Embedded Eye-Gaze Tracking On Mobile Devices

Stephen Ackland

Faculty of Technology

De Montfort University

A thesis submitted in partial fulfilment of the requirements of De Montfort University for the degree of

Doctor of Philosophy

April 2017

Acknowledgements

I would first like to thank the past and present members of my supervisory team, Dr. Howell Istance, Dr. Simon Coupland, Dr. Stephen Vickers and more recently Prof. Francisco Chiclana who have all played a role at one point or another in helping me reach the end of this long journey.

I would like to take this opportunity to thank my friends and colleagues working within the Centre for Computational Intelligence (CCI) and other departments at De Montfort University for their friendship, guidance and support when I needed it.

A special thanks goes to my mother Julie for believing in me and being my lifelong inspiration.

My greatest thanks goes to my wife Sian for offering endless love and encouragement. Without her I would not have had the strength to complete this work and for that I am eternally grateful. Last but not least I would like to thank my newborn son James for filling my life with joy every day.

Abstract

The eyes are one of the most expressive non-verbal tools a person has and they are able to communicate a great deal to the outside world about the intentions of that person. Being able to decipher these communications through robust and non-intrusive gaze tracking techniques is increasingly important as we look toward improving Human-Computer Interaction (HCI). Traditionally, devices which are able to determine a user's gaze are large, expensive and often restrictive. This work investigates the prospect of using common mobile devices such as tablets and phones as an alternative means for obtaining a user's gaze. Mobile devices now often contain high resolution cameras, and their ever increasing computational power allows increasingly complex algorithms to be performed in real time. A mobile solution allows us to turn that device into a dedicated portable gaze-tracking device for use in a wide variety of situations.

This work specifically looks at where the challenges lie in transitioning current state-of-the-art gaze methodologies to mobile devices and suggests novel solutions to counteract the specific challenges of the medium. In particular, when the mobile device is held in the hands fast changes in position and orientation of the user can occur. In addition, since these devices lack the technologies typically ubiquitous to gaze estimation such as infra-red lighting, novel alternatives are required that work under common everyday conditions.

A person's gaze can be determined through both their head pose as well as the orientation of the eye relative to the head. To meet the challenges outlined a geometric approach is taken where a new model for each is introduced that by design are completely synchronised through a common origin. First, a novel 3D head-pose estimation model called the 2.5D Constrained Local Model (2.5D CLM) is introduced that directly and reliably obtains the head-pose from a monocular camera. Then, a new model for gaze-estimation is introduced – the

Constrained Geometric Binocular Model (CGBM), where the visual ray representing the gaze from each eye is jointly optimised to intersect a known monitor plane in 3D space. The potential for both is that the burden of calibration is placed on the camera and monitor setup, which on mobile devices are fixed and can be determined during factory construction. In turn, the user requires either no calibration or optionally a one-time estimation of the visual offset angle. This work details the new models and specifically investigates their applicability and suitability in terms of their potential to be used on mobile platforms.

Contents

List of Figures

List of Tables

Glossary

BU head tracking dataset A dataset for head-pose estimation recorded from a low resolution RGB camera.

***Camera* factors** Parameters such as *frames-per-second* (*fps*), resolution and focal length that affect the amount of useful pixel data available.

Columbia dataset A high resolution image dataset for gaze estimation.

DMUHAG dataset A dataset for head-pose and eye-gaze estimation recorded from a RGB-D camera on a tablet device.

***dynamic* device** A mobile device that is currently subject to motion e.g. held in a user's hands.

embedded Utilising only the on-board computational ability and hardware of the device.

***Environment* factors** Parameters affected by the placement of the camera such as lighting conditions, backgrounds and whether the camera is in motion or not.

EYEDIAP dataset A dataset for head-pose and eye-gaze estimation from remote RGB and RGB-D cameras.

'flock-of-birds' An electromagnetic tracker that captures 3D position an orientation data.

generalised Procrustes Statistical shape analysis to normalise shape data to an optimally determined mean reference shape.

Intel Realsense An RGB-D camera that utilises time-of-flight measurements to measure short distances.

Microsoft Kinect An RGB-D camera that utilises time-of-flight measurements to measure distance.

mobile device A device such as a laptop, tablet or mobile phone that can be used in a wide variety of different locations and is potentially subject to movement during use.

MultiPIE A large annotated image collection capturing people from synchronised cameras at varying angles.

optical axis an estimate of gaze direction defined on the line along the centre of the eyeball and the centre of the pupil.

real-time Processing the image data as quickly as it as acquired to provide minimal delay between the user input and corresponding output.

***static* device** A mobile device that is currently stationary e.g. placed on a table.

UnityEyes An application to generate high quality synthetic models of the eye-region.

UPNA (GI4E) head pose dataset A dataset for head-pose estimation recorded from a RGB camera and 'flock-of-birds' tracker.

***User* factors** Characteristics of the participant including gender, age and ethnicity as well as things that obscure facial features such as hair and the use of glasses.

visual axis an estimate of gaze direction defined on the line along the fovea and the nodal point on the lens.

Acronyms

fps *frames-per-second.*

***k*NN** k-Nearest Neighbour.

2.5D CLM 2.5D Constrained Local Model.

3DMM 3D Morphable Model.

AAM Active Appearance Model.

ALR Adaptive Linear Regression.

ANN Artificial Neural Network.

ASM Active Shape Model.

BU Boston University.

CGBM Constrained Geometric Binocular Model.

CHM Cylindrical Head Model.

CLM Constrained Local Model.

CNN Convolutional Neural Network.

DMUHAG De Montfort University Head And Gaze.

EM Expectation Maximisation.

HCI Human-Computer Interaction.

HD High Definition.

LED Light-Emitting Diode.

LNF Local Neural Field.

MAE Mean Angular Error.

MAP Maximum A-Posteriori.

ML Maximum-Likelihood.

MOSSE Minimum Output Sum of Squared Error.

NRSfM Non-Rigid Structure from Motion.

PCA Principal Component Analysis.

PDM Point Distribution Model.

PoR Point-of-Regard.

POSIT Pose from Orthography and Scaling with ITerations.

PSO Particle Swarm Optimisation.

PSR Peak to Sidelobe Ratio.

RANSAC Random Sample Consensus.

RF Random Forests.

RGB Red, Green and Blue.

RGB-D Red, Green, Blue and Depth.

SD Standard Definition.

SDK Software Development Kit.

SVD Singular Value Decomposition.

SVM Support Vector Machine.

SVR Support Vector Regressor.

UPNA Public University of Navarre.

VGA Video Graphics Array.

Part I

Introduction and Background

Chapter 1

Introduction

The eyes are one of the most expressive non-verbal tools a person has and they are able to communicate a great deal to the outside world about the intentions of that person. Being able to decipher these communications through robust and non-intrusive gaze tracking techniques is increasingly important as we look toward improving Human-Computer Interaction (HCI).

This thesis investigates the prospect of incorporating gaze tracking capabilities into mobile devices without modification, particularly laptops, tablets and smart phones. That is to say, the processing power and memory available on such devices now allows for computationally intensive applications to run in real-time, and coupled with the inclusion of high resolution cameras as if often the case, there is a strong potential to develop a portable dedicated eye tracking device without making any additional alterations to the device itself. Gaze tracking on such devices could provide another modality of input instead of or alongside touch interfaces that have become ubiquitous on such platforms.

1.1 Motivation

Eye trackers are sophisticated pieces of equipment used in a number of different research scenarios to acquire an estimate of where a user is looking. They are also frequently relied upon by people with disabilities where the ability to interact with a computer in an alternative way can have a large social and positive psychological

impact. They can be prohibitively expensive, although there is a recent trend in producing lower cost devices that make some sacrifices in terms of accuracy and robustness. Typically, these devices require the use of multiple high resolution cameras and/or infra-red lighting to determine the gaze from a user.

This work seeks to investigate alternative approaches that have the potential to be embedded within standard mobile devices without modification. Such algorithms need to work using only a single on-board camera and must forgo the use of infra-red, working under only natural light. As the computer power available on mobile devices increases, building an eye and gaze tracker on them becomes an alternative and increasingly feasible solution and is the subject of recent research (Huang et al., 2015; Wood and Bulling, 2014). The high-resolution cameras that are normally included in modern devices such as mobile phones, laptops and tablets have increased the potential of lifting the current restrictions on gaze tracking. Such devices could take advantage of such an input modality to offer intelligent user interfaces that decrease the over-reliance on touch inputs and maximise screen real-estate. More promisingly, future work may utilise the technology in more complex scenarios such as in an augmented reality device while in natural environments, away from controlled lab conditions. Developments in mobile eye-tracking research may lead to an unobtrusive and reliable human-computer interface that can be used in everyday settings by a variety of people. Whether a sequence of images from a static or mobile camera, the gaze direction can provide important information about a user's intentions. Once we know where a person is looking, this new paradigm of mobile interaction can be combined with standard touch and motion interactions. Other application areas include hands-free situations such as whilst driving.

1.2 Existing strategies for mobile devices

While research into gaze tracking has been ongoing for some time, research for eye tracking on mobile devices is still relatively young. The literature has shown that there are various scientific and technical challenges to overcome in achieving mobile gaze-tracking, particularly in an application's ability to compensate for variability in positioning and natural lighting (Hansen and Ji, 2010). For smaller 'phone' screens Nagamatsu, Yamamoto and Sato (2010) and Miluzzo, Wang and

Campbell (2010) developed 'MobiGaze' and 'Eyephone' – two early examples of an attempt to interpret eye data from a phone, with both showing major restrictions. MobiGaze provided a gaze interface with touch-screen input to reduce the Midas touch problem. An application they developed allowed users to specify an area of a map by gaze and use their hands to zoom in on that area. However, the authors required the additional use of an infra-red LED and stereo camera attachments to generate a stream of gaze data, resulting in a very bulky prototype. The image processing also had to be carried out on a separate device, although this is less likely to be the case on more modern equipment. Eyephone tracks a user's eyes to activate different applications on the phone. The display is split into a three by three grid of options for a user to choose and activate by winking. In practice winking is not an ideal interaction modality and a grid of three options is severely limited.

In a study by Drewes et al. (2007), gaze gestures (whereby the software is manipulated via sweeping movements) are compared to more traditional position tracking algorithms on mobile phones. The authors suggest that it provides a viable alternative for mobile phones as the screen space for positional tracking on a mobile is small, and also that a user is unlikely to perform a gesture unwittingly. Additionally, gestures are more robust and do not require as thorough a calibration process (since the point of regard is not required), which is potentially more suitable for a mobile environment. However, a qualitative study comparing the techniques by the authors suggested that users considered the positional tracking interface to be more intuitive than its gesture counterpart. More recently, and on modern hardware, a similar methodology was taken in 'GazeSpeak' (Zhang et al., 2017) with gaze gestures used to good effect in real-time to facilitate communication for people with motor disabilities. It is able to represent 6 gestures for each eye that correspond to the user looking in various directions, looking centrally and blinking. The calibration procedure involves taking cropped images of each eye while performing the gestures and normalising them by utilising tracked points on the head. The calibration images are stored on the device and can be compared with new incoming data to estimate the current gaze gesture.

Alternative methods have shown to be viable on larger display screens such as tablets and laptops (Figure 1.1). Holland and Komogortsev (2012) used Artificial Neural Networks (ANNs) to obtain some encouraging results using an unmodi-

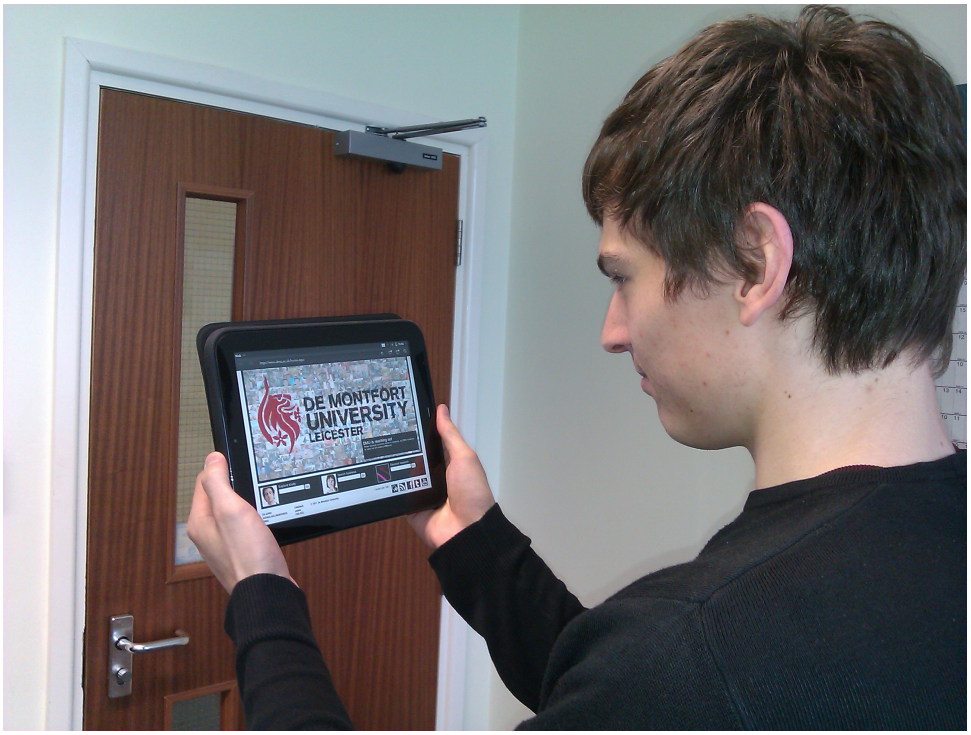


Fig. 1.1 Tablets offer a larger screen display than phones, although they can be more cumbersome to hold for long periods.

fied tablet computer. They use a regression model approach based on appearance with the individual pixel values fed into the network to generate gaze coordinates. The accuracy of the system is limited, allowing only to detect which quarter of the screen the user was looking at. Other limitations include the lack of head and tablet movement the system can tolerate, effectively removing its usefulness in mobile settings. Additionally, calibration is uncomfortable and very slow (taking approx 5 minutes of user interaction), and needs to be done at least once per usage/ per person, particularly if lighting conditions are changed. Alternatively, Wood and Bulling (2014) describe the 'EyeTab' system that estimates the iris centres on a commodity tablet device by using an objective function involving gradient images. They robustly fit an ellipse to the iris with a Random Sample Consensus (RANSAC (Fischler and Bolles, 1981)) approach to remove outliers, but this was found to cause quite a bottleneck on a relatively low powered device. They were able to achieve 7 degrees accuracy at 12 *frames-per-second* (*fps*), however the test participants were instructed to keep their head at 20cm distance away from the tablet device which is unrealistic in practice. It is suggested that

this may not have been required if the on-board camera was of a higher resolution than the 1280x720 pixel camera used in the study. Karamchandani et al. (2015) demonstrate a simpler setup on a tablet for children with disabilities. Their work focuses on utilising threshold techniques to estimate the iris centre to determine gaze on a 4x4 grid display. The system suffered however from noise from motion and illumination changes.

One of the problems in developing novel person-independent gaze interaction methods on mobile devices is acquiring sufficient amounts of training data to learn from. To overcome this issue Krafka et al. (2016) utilised crowdsourcing using a public mobile application called 'GazeCapture' that automatically collected images from over 1450 people giving over 2.5 million gaze samples. This large dataset allowed them to train a deep Convolutional Neural Network (CNN) to predict gaze at 10–15 *frames-per-second* (*fps*) on a mobile device. However, most of the training data is captured on a specific make of mobile phone and is shown to perform worse on tablet devices, which indicates the method is not independent of device or even the orientation in which the device is held. It would therefore likely benefit from separate networks being trained for different devices and orientations.

1.3 Aims and objectives

The general aims of the research are outlined below, along with a number of objectives that the thesis will be structured around in order to achieve these aims.

Aims

A1 To design a framework for determining a user's eye gaze under conditions that allow for head movement.

A2 To develop novel eye gaze interaction methodologies suitable for real-time use on mobile devices.

Objectives

O1 To carry out an extensive review of the literature in order to identify the current state-of-the-art within eye gaze research and to determine the limiting factors in bringing the methodologies to a mobile platform under free-head movement.

O2 To construct models of the head and eye that are synchronised and designed specifically within a gaze framework that can be directly applied on mobile devices.

O3 To construct a dataset of people using a mobile device collected from the device itself under *static* and *dynamic* conditions.

O4 To test the feasibility of the gaze model, validating its accuracy within a variety of contexts and subsequently identifying all areas within a gaze interaction framework which impact on the models accuracy.

1.4 Research approach

It is known that the two main influencing factors for the direction of eye-gaze are head pose and eye rotation relative to the head (Langton et al., 2004). Many techniques simplify the problem by only considering one and fixing the other, i.e. assuming the head position remains relatively stationary and creating a 2D mapping of the eye position to screen coordinates via a simple calibration procedure, or foregoing the eye location and simply determine the gaze vector via the orientation of the head.

One of the most common 2D mappings uses infra-red technologies through the well-documented pupil-glint vector (Merchant et al., 1974). This infra-red light source is invisible to the user making its use far more comfortable than using natural light. Infra-red additionally provides a quick way of determining the eye location through the bright/dark pupil technique, whereby the light source reflecting back off the user's retina is easily picked up within an image.

Infra-red light sources are not typically available on mobile devices, and their inclusion would not necessarily create a viable solution within the unrestricted

'mobile' domain. This is because, by definition, mobile cameras are not static, and therefore calibration procedures would likely need to be performed too frequently to be useful to the average user. Perhaps less importantly, interference from infra-red light from the sun would restrict outdoor usage of the technology. As such, alternative methodologies that would allow gaze interaction without the need for further modification of the device have to be found.

To overcome these issues it seems that we must look towards gaining knowledge about both fundamental aspects of eye gaze, i.e. determining the orientation of the head and subsequently combining knowledge gained from the relative eye locations and orientations. Recent trends seen in the literature that have shown good results for general purpose eye-gaze estimation utilise appearance-based methods where cropped out image of the eyes are rectified to a standard training position using a head-pose estimate. Machine learning strategies are applied to a large number of training samples to train a model that can evaluate the gaze from a new set of image pixels. This avoids the need to explicitly determine features of the eye such as the iris and pupil which are significantly more difficult to obtain when infra-red is not used. While this is an effective strategy, the biological functions of the eyes are well-understood and alternative feature and model-based methods are likely to perform just as well provided the features of the eye can reliably be estimated.

This work seeks to tightly couple new models for head-pose and eye-gaze estimation to carefully extract image features within the data provided by the forward-facing camera on a mobile device. A general purpose geometric model will be implemented that pushes the burden of calibration onto the camera and screen, and requires either no or one-time calibration from the user.

1.5 Research questions

The main research question addressed by this thesis is:

Can a reliable stream of gaze data be generated from a mobile device?

To answer this question, several objectives are to be met. Firstly, the suitability of current state-of-the-art eye-gaze estimation strategies are to be evaluated:

Which current algorithms can be adapted and embedded to run in real-time on an unmodified mobile device?

Assuming a stream of gaze data can be generated, what are the limiting factors and how much do they influence?

Secondly, there is a need to develop a method of tracking the head and the eyes within the camera image to accommodate the 'mobile' aspect of the problem:

Can we track the head (and in turn the eyes) quickly and accurately enough with limited computational power to collect gaze data?

How does the change in the user's head pose affect our ability to estimate gaze?

Finally, there is a need to determine the need for calibration, if any, for individual users:

Which parameters of the framework need to be calibrated for individuals and how often does this need to take place?

1.6 Contributions

The first main contribution of this thesis involves the creation of the De Montfort University Head And Gaze (DMUHAG) dataset, specifically for evaluating head pose and gaze estimation on a mobile device. Eleven participants took part in the recordings with their head shape being scanned to obtain their 'real-3D' face shape. Furthermore, two videos were recorded for each user on an RGB-D camera where the participants were asked to look at a series of targets on the mobile device in two modes – first with the tablet *staticly* placed on a desk and secondly, in a *dynamic* situation where the tablet was held in the hands.

The second contribution involves the design and development of the 2.5D Constrained Local Model (2.5D CLM) for head pose estimation from a monocular

camera. The model combines a 3D shape with 2D texture information that provides *direct* estimation of the pose parameters, avoiding the need for additional optimisation strategies. The model is carefully constructed with the alignment of the training shapes via the inner eye-corners, reducing the complexity of the model and allowing for simple synchronisation with the position of the eyes. A number of small enhancements are also designed and implemented, including a heuristic that globally aligns the face shape consistently and the optimisation of the texture filters to reduce their computational complexity. The model is complete with an analytical derivation of a Jacobian matrix, which describes how changes in the parameters of the model create changes in the point positions within the image through a full-perspective camera model.

The third main contribution is the introduction of the Constrained Geometric Binocular Model (CGBM) – an implementation of a geometric model for eye-gaze that combines a 3D shape point model with 2D texture information. Like the head-pose model, it also provides *direct* estimation of the pose parameters and the screen coordinates simultaneously. The shape model is binocular and symbiotically links the head-pose and eye-gaze models through a shared pose and local origin. The eye shapes and gaze direction are evaluated through a multi-dimensional and multivariate eye-orientation manifold. Finally, a thorough derivation of the Jacobian matrix is determined that describes how changes in the screen coordinates create changes in the point positions within the image.

1.7 Report structure

This thesis is structured into five main parts, (see Figure 1.2). The locations where the objectives (**O1-O4** from section 1.3) are met are shown where appropriate.

Part I: Introduction and Background

This part presents the motivation for the research and defines the research questions. *Chapter 2* discusses the state-of-the-art within the field of eye-gaze estimation and the difficulties in adopting these strategies for current unmodified mobile

systems. This leads to a discussion of the literature regarding 3D head-pose estimation from a monocular camera in real-time and potential issues that occur within a mobile setup. The section concludes with a review which discusses potential avenues for computationally inexpensive integration of 3D head-pose tracking with eye-gaze estimation (**O1**).

Part II: Research Methodology

Chapter 3 describes the datasets that will be used to train and test the new models. Additionally it describes how a small dataset (DMUHAG) was collected that attempts to isolate the conditions for holding a tablet device in the hands (**O3**).

Part III: Proposed Solution

The third part of the work contains the proposed solution with new models for head-pose and eye-gaze estimation on mobile devices (**O2**). It defines how the models are symbiotically linked though a common origin and pose parameters. *Chapter 4* introduces a model for 3D head pose estimation on mobile devices called the 2.5D Constrained Local Model (2.5D CLM), which utilises a 3D shape and 2D texture models to create an efficient and robust tracker. *Chapter 5* introduces the Constrained Geometric Binocular Model (CGBM) whereby features of the eyelids and iris are detected within the image and the gaze estimate is determined through a multi-dimensional manifold that optimises the 3D feature points of both eyes simultaneously to determine the gaze vector intersect on the calibrated monitor plane.

Part IV: Experimental Evaluation

Chapter 6 Provides the results of the experiments performed on a number of datasets and contains an evaluation process that attempts to draw out the key benefits and weaknesses of the newly proposed solution (**O4**).

Part V: Conclusions and Future Research

Finally, *Chapter 7* summarises the content of the thesis and seeks to provide answers to the research questions. Additionally, it provides a discussion of multiple possible research avenues that arise from the experimental results.

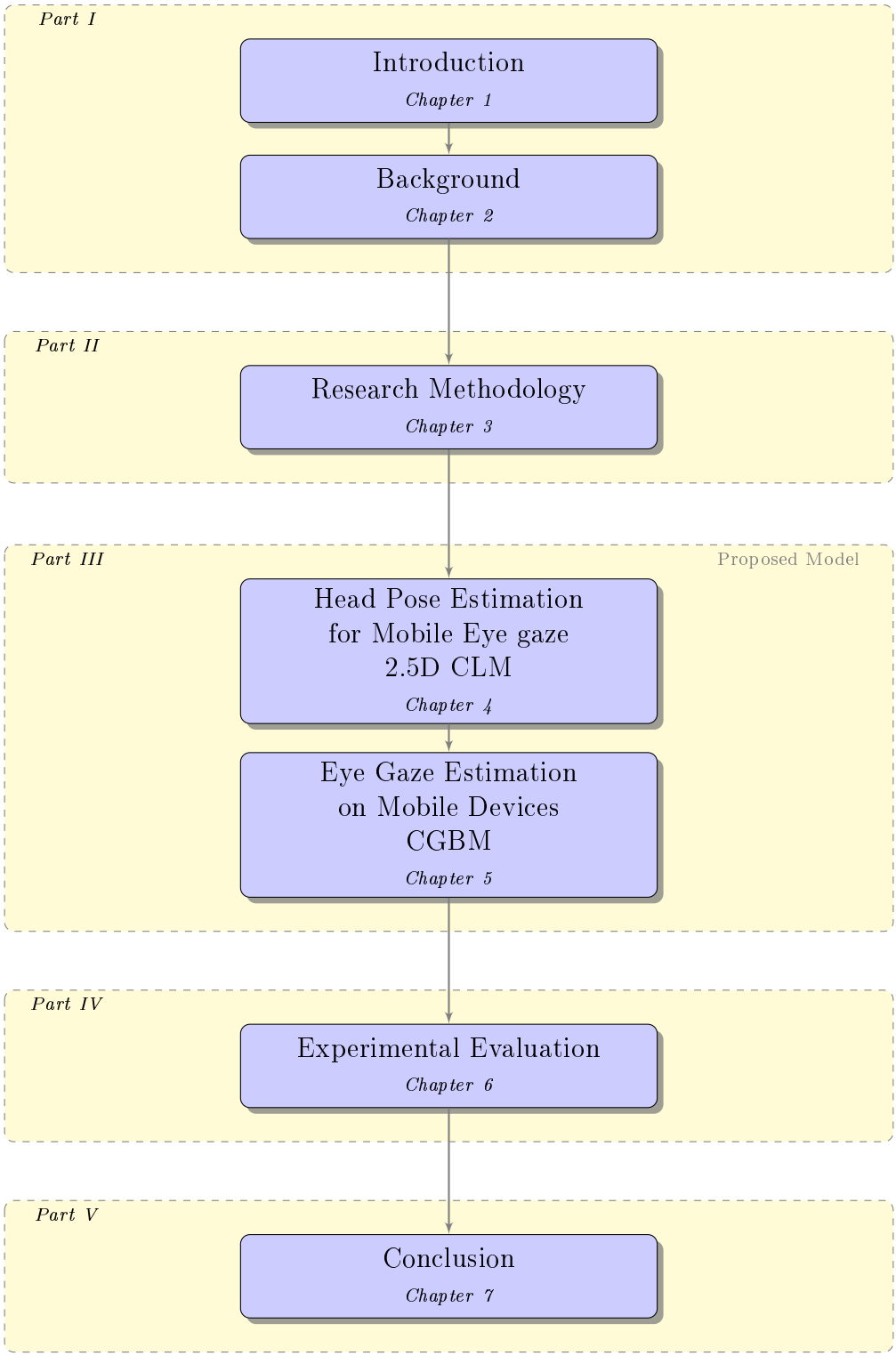


Fig. 1.2 Structure of the thesis

Chapter 2

Background

The aim of this chapter is to give an overview of the symbiotic relationship between the head and the eyes. It begins with a brief introduction of the internals of the eye that we need to observe to understand gaze estimation strategies. From there, we can understand the two main approaches to gaze estimation – interpolation-based and geometric-based approaches. The chapter then goes on to make the case for determining and adding the head pose to the calculations for both gaze approaches, and discusses strategies for firstly detecting the head within the image and in turn approximating the actual head pose (rotation and translation values). Finally, the chapter is concluded with a discussion on the considerations to be made for eye-gaze to be used on mobile devices as well as the calibration required to perform these techniques.

2.1 The eye

The internals of the eye have long been studied and are well understood for the purposes of eye gaze estimation (Duchowski, 2007; Huey, 1908). The outermost layer of the eye (sometimes known as the white of the eye) is called the sclera (see Figure 2.1). At the front part of the eye and continuous with the sclera is the cornea, a convex and transparent region that protrudes slightly such that the overall eye shape is no longer spherical. The cornea is the first point of contact with light entering the eye and helps refract it towards the retina. Behind the

cornea is the pairing of the iris and pupil – the iris being the coloured circular part of the eye that dilates and constricts to adjust the size of the disc shaped hole at its centre (the pupil). The larger the pupil, the more light is allowed to enter into the eye.

Behind the iris is the lens, a bi-convex transparent disc, that can be adjusted and flexed using the muscles within the eye. The purpose of this is to further focus the light on the retina (the lining at the back of the eye). The retina is made up of a dense collection of photoreceptor cells called rods and cones which produce our visual field from the areas of focus to the periphery. Rods are sensitive to low lighting and monochrome, whereas cones are adept at coloured light. Within the retina is a very small region called the fovea which has a very large number of cones in order to produce a clear image of our surroundings. At the very back of the eye is the optic nerve which carries the impulses detected on the retina to the brain.

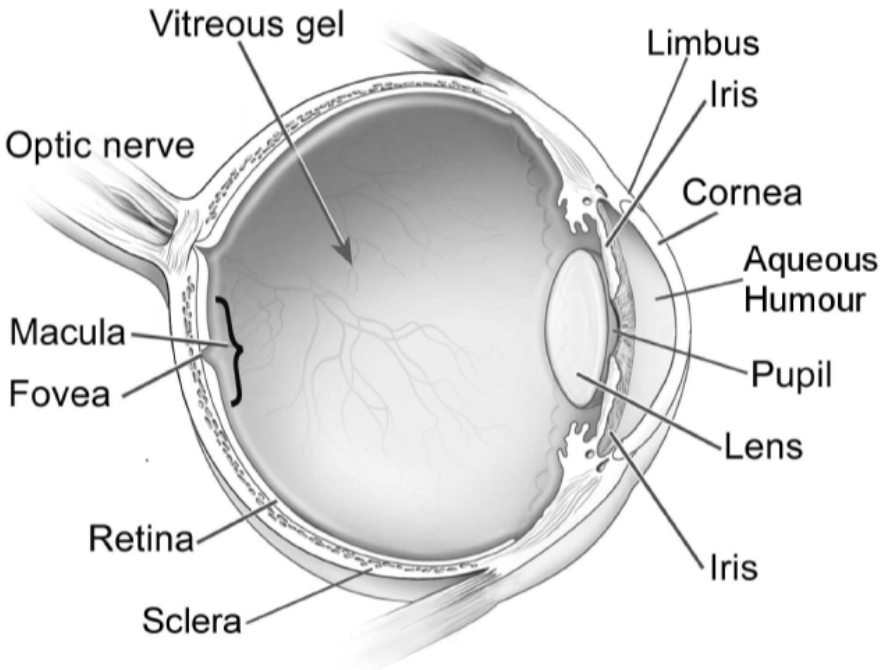


Fig. 2.1 Diagram of the eye. (National-Eye-Institute, 2013)

A person's gaze can be interpreted as both their gaze direction and also the Point-of-Regard (PoR) that represents the 3D point-location that is being fo-

cussed upon. The gaze direction then is the 3D direction vector between the eye and the PoR. For the purposes of Human-Computer Interaction (HCI), the PoR can also have a 2D representation as a set of gaze coordinates on a monitor or alternative screen display. The optical axis is the line that connects the centres of the pupil and eyeball itself, and may initially appear to be the most obvious choice for defining the direction of a person's gaze. However, the visual axis (defined as the line connecting the fovea to the nodal point on the eye lens – the centre of corneal curvature) is often determined to be the true line of sight due to the high concentration of cones at the fovea, and for a typical adult is approximately 4-5 degrees offset from the optical axis at the nodal point (see Figure 2.2). The angle offset κ remains static for the majority of a person's life and can be determined via a one-time calibration procedure. Models attempting to derive additional elements such as the radius of the cornea at its apex and refraction parameters may further help to determine the true line of sight (Guestrin and Eizenman, 2006).

There are many examples in the literature of modelling the eye to varying complexities in order to evaluate the gaze direction or the PoR directly. The following sections will describe the common approaches taken.

2.2 Gaze estimation

Gaze estimation methods usually fall into one of two varieties: the model-based (geometric) approach and the interpolation-based (regression) approach (Hansen and Ji, 2010). Interpolation-based approaches typically form some kind of mapping from the image to gaze coordinates, either from features directly detected in the image such as the glint and other reflections on the cornea (known as Purkinje reflections), or from the appearance of the image pixels directly. The geometric approach focuses rather on the direct 3D estimation of the optical or visual axis from the eye to the PoR via detected image features such as the iris and pupil. Either method can be subject to a number of different calibration procedures for the individual user and the environment setup. For example, the regression parameters for the user estimated once per session or a one-time estimate of the visual axis offset. The environment calibration can include amongst other things, the position of the camera, monitor and any external lights utilised. The pa-

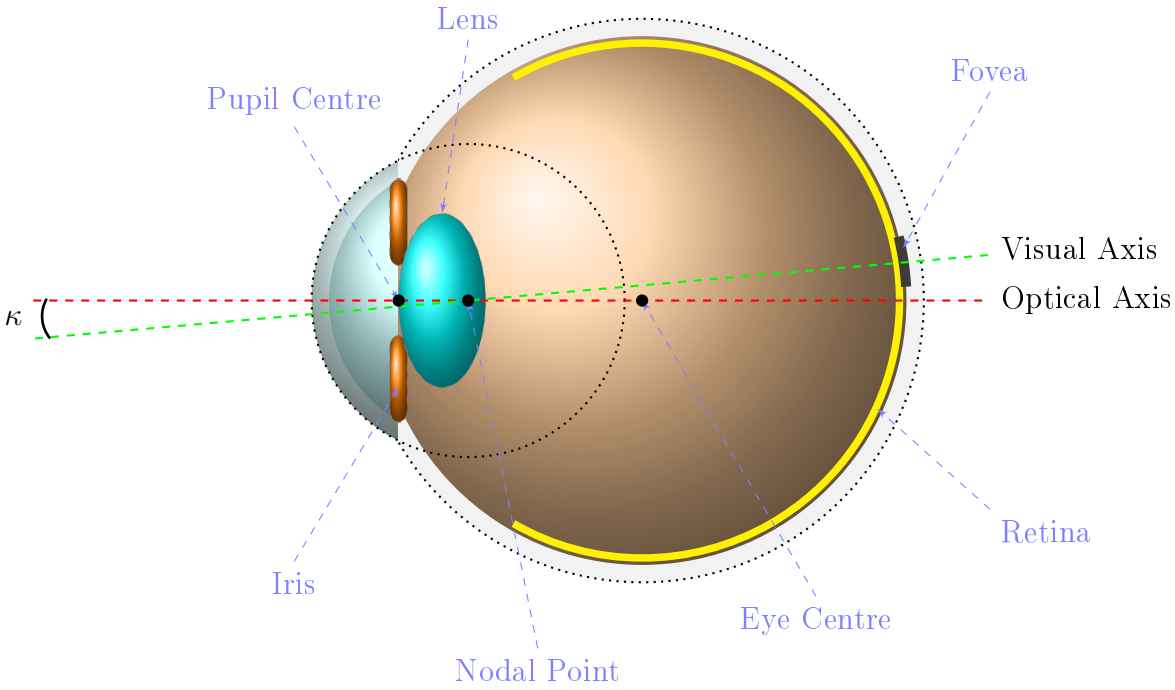


Fig. 2.2 Diagram of the right eye optical and visual axes. A person's gaze can be modelled as a vector along these axes.

rameters that require calibration are solution-specific and will be commented on throughout this chapter.

For mobile eye-gaze to be used in everyday tasks, normal use of the device should attempt to mitigate the need for (re)calibration as much as possible for the user. Calibration procedures often ask the user to look at specific points of interest (such as following an on-screen circle, see Figure 2.3) which allows the model to acquire the ground truth of the user's gaze. The number of targets that are required to be observed is directly related to the complexity of the model and how many parameters there are to estimate. For example, under geometric methods estimating the angle offset between the visual and optical axes may only require one calibration target whereas an interpolation-based approach such as polynomial regression would require a larger number of targets spread across the monitor. To ensure the calibration is performed correctly the process is performed at a speed that is comfortable to the user, which typically involves collecting a small number of user fixations. It is a fairly rigid procedure that can be monotonous if it is needed to be performed too often, even for short calibration

procedures. Additionally, depending on the method used a number of other calibration parameters may be required that are not dependent on the user, such as the calibration of the camera and the position of the monitor relative to the camera. While obtaining some of these parameters can be quite complex initially, it may allow for the number of user parameters required to be reduced, or the calibration to be performed less frequently. An important side-note to consider is whether some of the parameters of the eye can be picked up at all and may need to be estimated. For example people who wear glasses (particularly those with strong lenses) or those who have droopy eyelids may make it difficult to robustly track certain eye features such as the iris (Daugman, 2007). The requirement of calibration is therefore of paramount importance, since if a methodology is accurate but requires constant maintenance then the users become frustrated from degrading performance and the overall usefulness of the approach is significantly impaired.

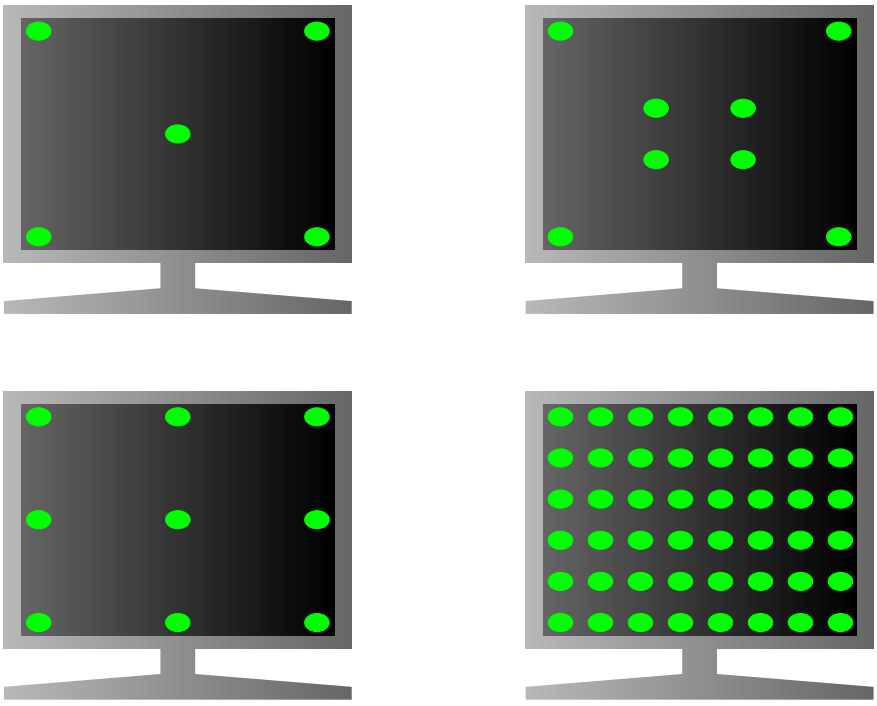


Fig. 2.3 Typical mapping calibration point strategies. The number of points required is relative to the complexity of the model.

Calibration processes can sometimes be streamlined by using a model of visual attention which builds a saliency map representing targets that the user is likely to focus upon. Evaluating the expected saliency against the estimated saliency,

we may be able to continuously update the parameters of our model. The eyes are actually capable of performing many fixations in a short period of time with average fixation times at 100ms – 300ms (Snowden et al., 2012) and the large volume of data this provides could potentially be exploited. Model and Eizenman (2012) demonstrate a probabilistic approach for determining the angle between the optical and visual axes (angle kappa) in a 3D eye model in infants. By utilising the knowledge that infants tend to look at patterned stimuli on a uniform background (Fantz et al., 1975), they were able to estimate the angle with less than 1% of false detection even though the babies only attended to the target image 47% of the time on average.

By comparing image saliency with the expected gaze of a 3D eye model, Chen and Ji (2011) overcome the need for active calibration from a user with a Bayesian probabilistic approach. The user's gaze estimation (specifically, the angle kappa) is improved gradually while viewing a sequence of images on the screen with known expected saliency. A similar approach was earlier proposed by Sugano et al. (2010) for short video sequences to update 2D mapping parameters via Gaussian regression.

2.2.1 Measuring accuracy of eye-gaze estimates

The standard for eye gaze accuracy measurement is the angle in degrees between the real gaze direction and the estimated gaze direction. Let $\mathbf{r}_g$ be the real gaze vector from the eye to the Point-of-Regard (PoR) and $\mathbf{e}_g$ be the estimated gaze vector. Then the angular error β is defined

$$\beta = \arccos\left(\frac{\mathbf{r}_g \bullet \mathbf{e}_g}{\|\mathbf{r}_g\| \, \|\mathbf{e}_g\|}\right) \tag{2.1}$$

Occasionally the error is given as a pixel-error on a monitor display, although the visual angle is preferred as it is independent of the screen size and resolution. If additional information about the scene setup is known, the angle can still be determined. Taking the Mean Angular Error (MAE) of all tested samples gives an overall picture of how accurate the gaze-estimation technique is.The accuracy measurement is heavily dependent on the distance between the user's eye and the PoR, where all else being equal the error term between the estimated and

real PoR decreases as the user moves further away from it. On the other hand, it becomes more difficult to observe the features of the eye as the user moves further away from the camera since there are fewer pixels that represent the eye region.

Test data is usually acquired through users observing targets, with the assumption that the user is looking precisely at the given target. Of course it is a reasonable assumption that this data is itself noisy since the visual field of a user is an area rather than a single point and therefore it it possible that the participant has the target within their sight, but are not fixating on its exact centre (a fixation itself being defined as a stable visual axis with a visual dispersion of less than 1° (Villanueva et al., 2008)). As such many estimation strategies involve a smoothing process, typically by averaging the estimations of gaze from each eye and also temporal averaging of many estimations over a short period of time (Huang et al., 2015).

Since acquiring a large volume of training and test data can be difficult, simulated data can be used in some cases. Böhme et al. (2008) released a MATLAB framework to compare different eye-tracking methods with simulated data. The framework gives the coordinates of relevant eye features in the image plane such as the pupil contour and glint. More recently Wood et al. (2016*b*) created a tool called UnityEyes which uses complex scanned 3D models of the eye region to generate lifelike synthesised images. The gaze vector can be easily adjusted to allow for testing and fine-tuning of new algorithms.

The following sections will explore a variety of interpolation and geometric-based gaze models to determine their suitability to work under the complex constraints of a mobile environment. When testing models, many works within the literature report a measure of accuracy, however it is important to recognise what constraints these systems were tested under. First the literature will be discussed and toward the end of this chapter a discussion will take place regarding the errors we can realistically expect under passive light conditions on a mobile device with pose, and ideally, even person invariance.

2.3 Interpolation-based approaches

Interpolation-based approaches attempt to build a regression from some input feature space defined within an incoming image, to screen coordinates. As such they can be and are often performed with no or minimal calibration of the camera and monitor, with each user instead viewing a selection of targets to provide the dependent and independent variables for regression. This is a key characteristic and benefit of the interpolation-based methods as they are relatively straightforward to setup and implement.

The most commonly used gaze estimation approaches are based on the use of infra-red light sources (Coutinho and Morimoto, 2006; Ebisawa, 2009), and use similar methods to the very earliest eye detection methods (Merchant et al., 1974). Infra-red firstly is comfortable for a user as the infra-red spectrum is not visible to the eye and allows for the quick detection of the eyes in an image via the bright/dark pupil effect that is produced. When an infra-red LED is placed in line with the eye and camera the light ray reflects off the retina producing the bright pupil effect within the image. Alternatively, a bright/dark response can be obtained by turning on and off two or more infra-red LEDsplaced at different locations with respect to the camera.

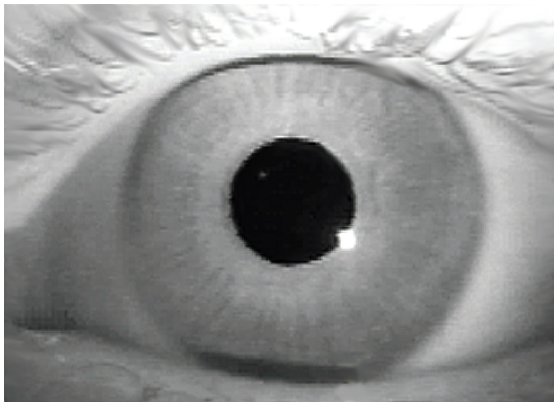


Fig. 2.4 The pupil and glint

The infra-red light source that is shone into the eye produces a number of detectable reflections off of the cornea and lens known as Purkinje reflections (Duchowski, 2007). The largest most easily detectable Purkinje reflection is called the glint and by utilising its position the gaze vector can be obtained (Figure 2.4). This is often achieved through a 2D mapping from the input vector defined as the

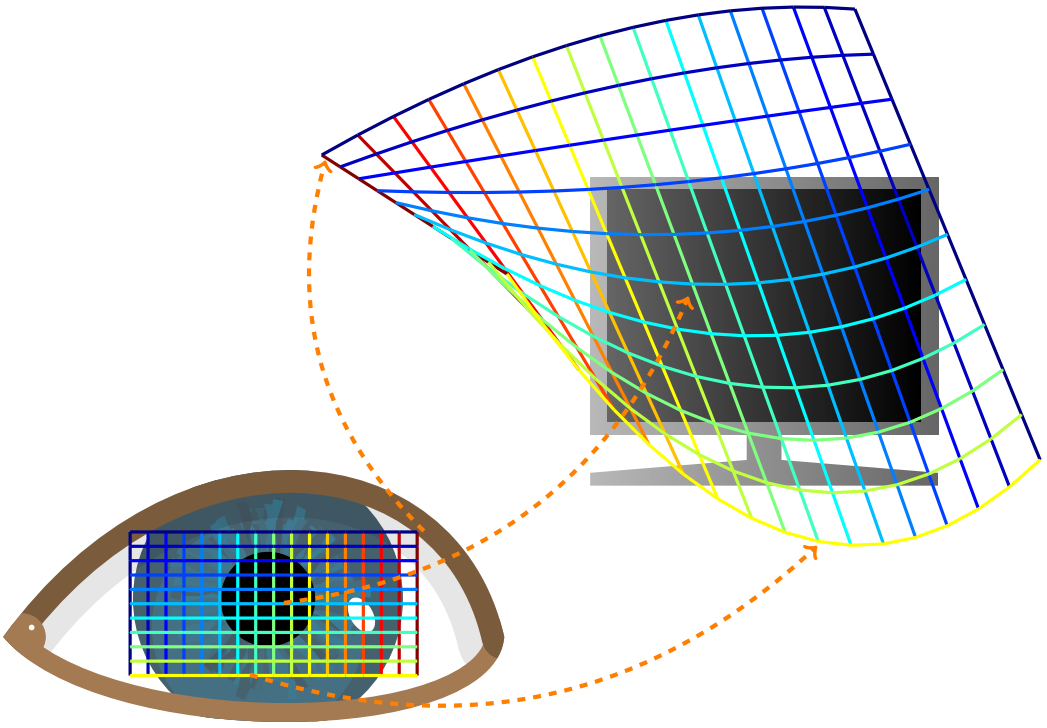


Fig. 2.5 A one-to-one mapping between the eye plane and the monitor plane. Note the instability of the mapping outside of the monitor region. The current location of the pupil centre maps to a specific location on the monitor plane

difference between the pupil centre and glint, to the output as screen coordinates (Figure 2.5). To build the mapping a regression approach can be undertaken by collecting training data from the user. It has been shown that a second-order polynomial is sufficient to approximate the user's gaze, and no additional benefit is typically gained from more complex approximations (Guestrin and Eizenman, 2006). Alternatively, detected feature locations such as the iris centre and eye corners as well as an estimate of the head pose can be inputs to an artificial neural network (Torricelli et al., 2008).

2.3.1 Interpolation-based approaches under passive light

Recently, attempts have been made to build an interpolation-based method under visible-light conditions by mapping between screen coordinates and the vector defined as the eye-corner to pupil centre (Shao et al., 2010), however there is still the issue that the mapping changes under head motion. While the eye corner may initially appear to be a stable anchor point, it has been shown that non-negligible inaccuracies are incurred due to a displacement of the eye corners when the eyes rotate (Sesma et al., 2012). Table 2.1 shows how the input from the pupil-corner method differs from the pupil-glint and appearance-based methods.

Estimating the iris centre and eye corners under passive light conditions can be a difficult challenge. Standard approaches tend to include a variety of low level computer vision algorithms for shape and feature estimation. For the eye corner estimation examples include Canny filters (Villanueva et al., 2013), which outline strong image edges in order to determine eyelid intersection, and Harris corners (Zhou et al., 2011), which scores all image edge intersection using derivative images. Additional preparation steps such as removing eyelashes can sometimes help in this regard. Alternatively, (Skodras et al., 2015) have shown a framework which incorporates an anchor template image patch, with gaze estimate determined by how the iris centre moves in relation to the centre of it. They use a Lucas-Kanade tracking algorithm (Baker and Matthews, 2004) to determine the location of the region over time rather than attempting to estimate specific eye features, although they perform no compensation for the potential change in appearance of the patch due to head rotation or lighting fluctuations.

For the iris centres Hough circles (Torricelli et al., 2008), isophotes (curves connecting areas of equal brightness) (Valenti and Gevers, 2011) and gradient images (Timm and Barth, 2011) have all shown some success when the eye image is of sufficient quality. Under low resolution images or when only the sides of the iris is visible, these methods can often fail due to insufficient data. Iterative and stochastic fitting approaches using a mixture of feature and model-based approaches have been shown to alleviate some of these issues (Kim et al., 2015; Li et al., 2005). Hansen and Pece (2005) used a particle filter for Expectation Maximisation (EM) of contours in an attempt to determine the outline of the iris. Similarly Chen et al. (2013) implement a variable circle sector separability

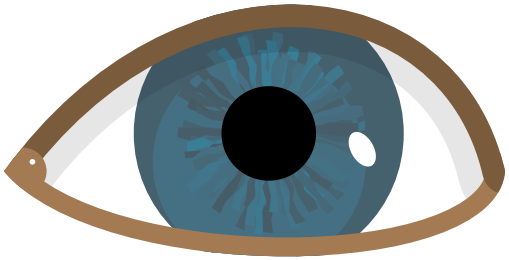
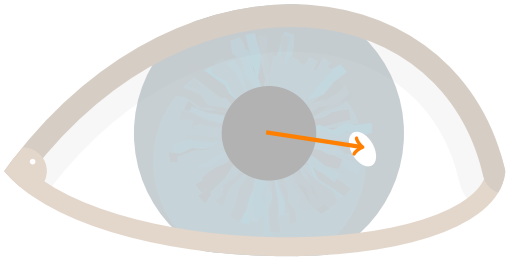
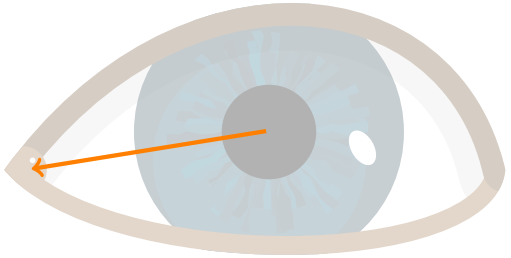
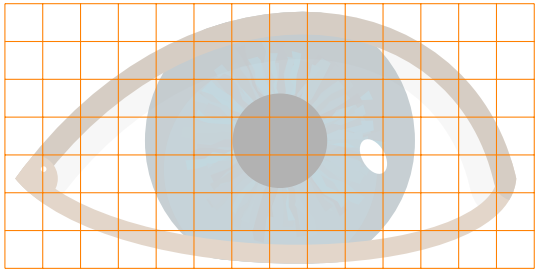
Incoming Image	Eye Gaze method	Input
	Pupil-Glint Vector	
	Pupil-Corner Vector	
	Appearance-Based	

Table 2.1 There are many ways of converting the incoming image data into an input for a mapping function. Estimating image features such as eye corners and glints allows us to extract simple vectors which lower the dimensionality of the problem. Alternatively, the individual pixels from the image can be fed directly into a mapping framework, potentially utilising more image information at the expense of higher computational complexity.

filter to detect iris outlines using a particle filter. The ranking of each particle is based on only the left and right side of iris since the eyelids bias the results. The minimum iris radius for tracking purposes was 10 pixels.

2.3.2 Pose-invariant regression approaches

Without the use of a sophisticated method to obtain the absolute head pose there has been only limited success in adapting the interpolation-based methods for use under free movement of the head. In uncalibrated camera setups, light sources can be placed at the 4 corners of a display in order to apply an invariant cross-ratio value in projective space on the corneal reflections (Coutinho and Morimoto, 2006; Yoo and Chung, 2005). This can be improved to better model the visual axis by performing a homography normalisation (Hansen et al., 2010, 2014). Lu et al. (2011) decomposed the problem into two parts; an initial estimation and compensation strategy based on geometric priors. They build up training samples by asking users to gaze at points while rotating their head, and using regression build a bias model using the incoming head pose. This model compensates for the change in appearance as the user rotates their head to give a more accurate gaze measure.

Cheung and Peng (2015) track the head movement and update the final gaze position by estimating the translation offset from the known calibration position. However, no adjustment is made for the head rotation or for the actual change in appearance of the eye region. Zhu and Ji (2007) take a more complex approach, where the gaze estimation of each eye is independently updated based on its own translation and rotation. The objective function is written in an iterative fashion, such that the two eye-estimations converge towards a single optimal result. Wang et al. (2009) use an image warping strategy by projecting a 3D ellipsoidal head model onto the image plane. The pixels lying on the 3D surface are then back-projected to a calibration pose so that the pupil-corner vector can be extracted, significantly reducing their errors.

There are several recent examples of other researchers in the field attempting to combine a head pose tracker and a gaze estimation model based on eye localisation. Valenti et al. (2012) use head-pose tracking with eye localisation cues to estimate gaze. A simple Cylindrical Head Model (CHM) is estimated and

tracked over time. Gaze estimation is determined via a head pose vector, and subsequently adapted via some calibrated 2D mapping values from the neutral pose eye centre. Feeding information from both the 3D head pose and eye localisation to each other allows for more accurate calculations from both. However, the head-tracker itself only determines head pose relative to some initial position and the tracking gradually degrades over time requiring constant reinitialisation and calibration. This simple combination provides good initial estimates of gaze direction demonstrating the usefulness of the approach, but the simplified head and eye model is limited in providing long-term accuracy and usability over time due to tracking errors.

One way to overcome the problems of regression-based approaches in regard to a change in head pose is to periodically update the regression parameters, although this pushes a heavy burden onto the user. Recently the 'WebGazer' model was made available for use online using common webcams (Papoutsaki et al., 2016). The model performs a regression with the pupil centre estimate and the local eye region to estimate the gaze. The novelty comes from the fact that it is able to continually update the parameters of the regression through the assumption that the user is looking at the location of the mouse cursor when the mouse is clicked. Since the regression is naturally updated, the model is able to adapt to new pose locations.

2.3.3 Appearance-based models

An alternative regression approach known as 'appearance-based' methods are carried out on the individual pixels representing the whole eye region directly rather than any specifically detected features. Sewell and Komogortsev (2010) used an unmodified laptop webcam to develop a real-time gaze-tracking system. The system feeds images of a user's face, eye and iris into an artificial neural network which is trained to estimate a user's gaze. The network is trained by asking the user to track 48 visual markers on the screen. For each marker, eight images of the eye are taken and represented in the neural network by the x and y coordinates of the screen. The authors note that this calibration process is too long to be used in real world scenarios, but could be potentially improved by pre-training the network using a large number of participants. Testing showed good

results although the success of the solution was heavily dependent on the pose of the user's head and were only suitable for use with the calibrated individual.

An interesting study by Lu and Chen (2015) developed a person-independent eye gaze prediction by utilising patch-based features of eye images. The person-independence of the more recent appearance have been achieved by utilising a large number of training data. Here a training set of eye images is used to build a 'codebook' comprising a set of basis patches. When a new eye image is obtained, the eye area is split into small patches from which they can rebuild the eye image, the result of which is already classified into one of 40 on-screen locations. No consideration is given however for the scene setup or head pose changes, limiting the technique's usefulness in real-world scenarios. Previously, they had demonstrated an Adaptive Linear Regression (ALR) appearance-based method which attempted to minimise the number of training samples required with some success (Lu et al., 2014).

The current state-of-the-art appearance-based models utilise a strong estimate of the absolute 3D head position in an attempt to make them pose-invariant. The pose is used to 'rectify' the image, that is to say, the image is transformed so as to appear as close as possible to the eye region under a training pose. The process for estimating the head pose (see section 2.6) to perform rectification typically involves additional calibration to further understand the relationship between the camera and user, although fully calibrated setups are not strictly required, unlike the geometric-based methods in section (2.4). In one such example, Funes-Mora and Odobez (2015) use the head pose acquired from an RGB-D camera to rectify the appearance of the eye to some canonical viewpoint. From there a number of different regression methods are applied on the image of the eye to obtain a 3D gaze vector. On a sample set of the EYEDIAP dataset (Mora et al., 2014) under person and pose invariance, angular errors were in the region of 7° with a Support Vector Regressor (SVR). For person and pose independence this can be considered very good although the use of depth-driven warping of the eye-region using a known 3D model of the head prevents the method from being used with traditional cameras. However, similar techniques that can be applied on RGB data have recently become very popular in the literature due their ability to perform well under the aforementioned pose and person-invariance restrictions. Such works include a k-Nearest Neighbour (kNN) approach (Wood et al., 2016*b*),

a Convolutional Neural Network (CNN) (Zhang et al., 2015), Random Forests (RF) (Sugano et al., 2014) and the Adaptive Linear Regression (ALR) method talked about previously (Lu et al., 2014) but adapted to use a larger number of training samples from a mix of different people (Zhang et al., 2015). All of these alternative models are tested on the EYEDIAP dataset using the head pose provided within the dataset itself as they do not have an explicit contribution for obtaining it naturally. Still, such methods could potentially be viable on a mobile device and since they have reported state-of-the-art results on an open dataset they are perfectly suited for use as a comparison metric for any newly proposed model. The results are discussed in more detail in section 2.5.

2.4 Geometric-based approaches

If the 3D geometric relationships between the eyes, camera and monitor can be obtained, then a 3D gaze vector can be estimated. Specifically, if we know the 3D pose of the eyes we can determine the lines of gaze represented by either the optical axis (that connects the centres of the pupil and eyeball) or the more complex visual axis and determine where they intersect the monitor plane. Typically both eyes are estimated independently and an average value taken. Figure 2.6 shows the convergence of such rays from the user's eyes to the screen from two different head poses.

For the gaze model in the geometric case to work, details about the environment parameters need to be known making their setup potentially restrictive and awkward (Hansen and Ji, 2010; Shih and Liu, 2004; Villanueva and Cabeza, 2007). Unlike the interpolation-based approaches, the geometric-based approaches strictly require the parameters of the environment setup to be fully calibrated. Such parameters include the 3D coordinates of the monitor corners with respect to the camera coordinate system (with the camera situated at the origin) and the camera intrinsic matrix which determines the focal length, principal point and skew parameter of the individual camera. These parameters are often obtained through image processing techniques using a checkerboard pattern (Weng et al., 1992).

This perceived cost of a more complex scene setup is offset by a simpler user

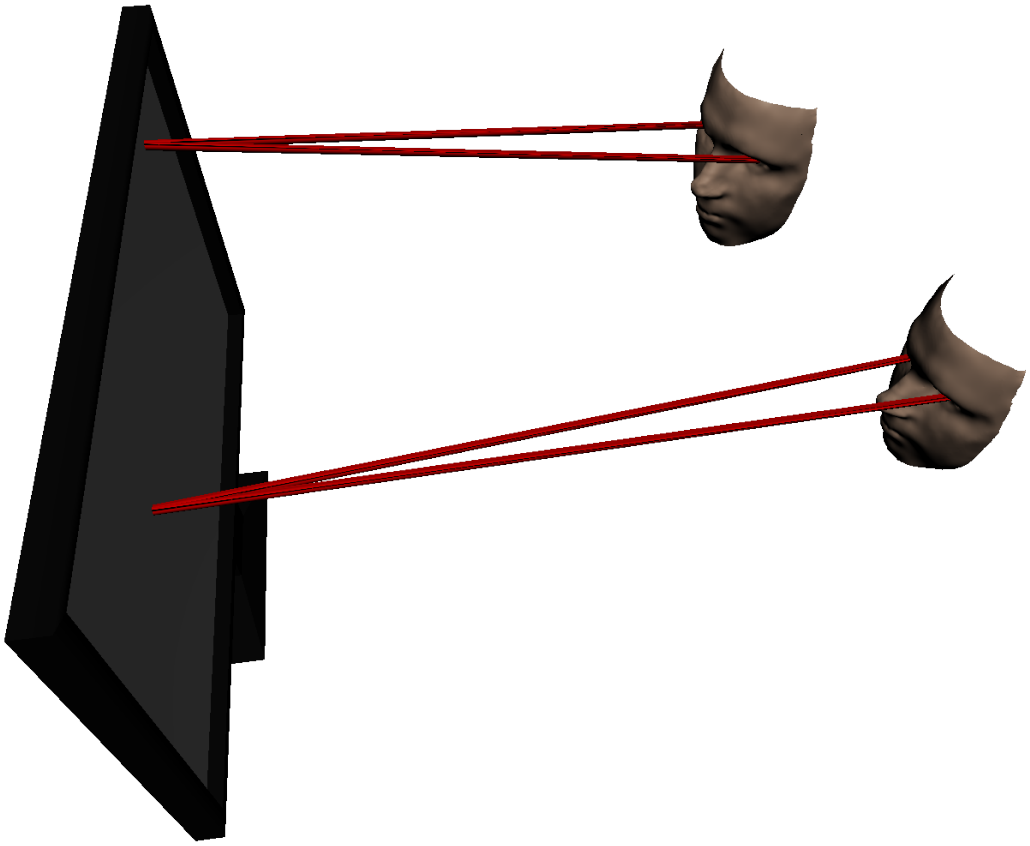


Fig. 2.6 Geometric methods attempt to utilise the pose of the head and/or the eyes, to directly evaluate the convergence of the visual or optical axis from the eyes upon the display.

calibration stage whereby the user's individual parameters may only need to be estimated once, and can be reused between completely different environments and scene setups. These user parameters can vary from the very simple to the complex depending on the choice of eye model. Typically a model may involve details about the person's eyeball shape, size, positions, the cornea shape and the visual axis angle offset κ. Any generalisations or simplifications may be the source of errors in the final estimation, but may be deemed acceptable under conditions where it is difficult to estimate the parameters. Luckily, the majority of these parameters are constant for an individual user over time (only potentially changing slightly over many years) and only need to be determined once (Berrio et al., 2010).

The optical and visual axes are related by the centre of corneal curvature (the

nodal point), making it one of the most important parameters to estimate. If the cornea curvature is known, a standard approach has been to use two infra-red light sources to estimate this point (Morimoto et al., 2002). Alternatively, solutions using a combination of single/multiple camera(s) and single/multiple light(s) have been evaluated (Guestrin and Eizenman, 2008). Fully calibrated stereo camera systems (Nagamatsu, Sugano, Iwamoto, Kamahara and Tanaka, 2010), head-mounted eye trackers (Kohlbecher et al., 2008) or restricting the user to place their head on a chin rest (Huang et al., 2010) are way of fixing the 3D eye centre locations without the need to estimate them explicitly.

If the 3D eye centres can instead be located via a robust 3D head pose tracker, we can obviate the need for infra-red light sources or stereo camera solutions. One such possibility for a successful geometric approach with a 3D head tracker would be to make some simplifications in that the eye is spherical and that the pupil or iris is actually a perfect circle residing at some 3D coordinate and orientation (see Figure 2.7). Pirri et al. (2011) were able to obtain an estimate of the line of gaze by first detecting the pupil ellipse within the image, and estimating its 3D pose. The gaze vector was then defined as the surface normal of this circle, which approximates the optical axis sufficiently. To demonstrate the effectiveness of this possibility, several works have utilised RGB-D cameras (i.e. colour with depth information) to provide reasonable estimates of the current head pose. Jianfeng and Shigang (2014) used the built-in algorithms provided by a *Microsoft Kinect* device to obtain the users head pose and utilised a 3D eyeball model to obtain the visual axis of gaze. The major factor for errors reported was from

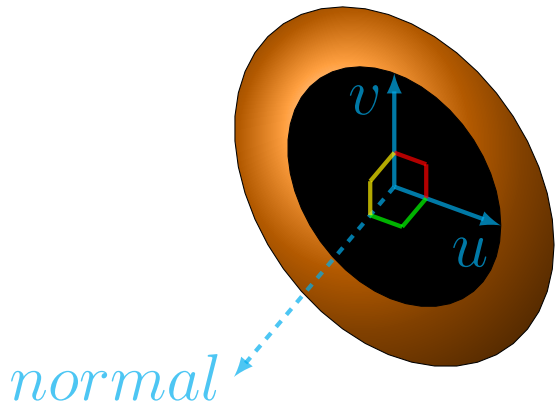


Fig. 2.7 Estimating the normal vector of a circle in 3D-space

the estimate of the pupil centre within the image where they used the method previously discussed of Timm and Barth (2011). Another work by Xiong et al. (2014) compares two approaches of obtaining the head pose. Firstly an RGB-D approach utilising the *Microsoft Kinect* and a RGB approach estimating head pose from a POSIT algorithm. A further improvement in using these 3D cameras is achieved by Funes-Mora and Odobez (2015), by utilising 3D Morphable Models (3DMMs) to generate a person specific face model in an offline phase. This allows them to track the head pose and in turn achieve consistent estimates of the eye locations. The following sections discuss head tracking strategies in more detail.

Other researchers are beginning to look at full 3D geometric solutions. Nakamatsu et al. (2012) use a 3D Active Appearance Model (AAM) of both the head and the eyes along with a stereo camera system. The research documents only limited results testing with a single user, and unfortunately the requirement for a highly complex setup with multiple cameras makes the work difficult to transfer across to mobile scenarios.

2.5 Accuracy expectations for gaze on a mobile device

While commercial and highly engineered setups can acquire accuracies less than 1° fixation dispersion, this is currently not realistic for a ubiquitous solution on a mobile device that is under the limitations of passive light and potentially large free head movement. Additionally since we are looking to minimise or remove completely the need for the user to calibrate, realistic benchmarks are to obtained by which we can determine the success or failure of a newly proposed model.

In general, many researchers test their models by utilising recorded camera data where the user is sat approximately 50–60cm away from the camera (Park, 2007; Skodras et al., 2015), considered a standard 'working distance' while sat at a desk. Since the requirements in the mobile context are so difficult researchers have taken to evaluating their models at considerably shorter distances. For example, the recent state-of-the-art pose and person invariant shape-based method 'EyeTab' (Wood and Bulling, 2014) evaluated their work at 20cm achieving errors in the region of 7°, with errors increasing as the resolution of the image data

around the eye region decreases. This enabled them to overcome the limitations of low quality eye-image data that would dramatically lower the accuracy and feasibility of using their method.

For testing in a mobile scenario Huang et al. (2015) collected an interesting dataset called 'TabletGaze' where users recorded videos from a number of different positions while looking at a sequence of gaze locations on a tablet device. The positions recorded included 'Lying Down' and 'Standing', which in many cases made the user only partially observable and sometimes leave the field of view of the camera entirely. In fact in only 30.8% of the videos was the entire face region visible which may lead to issues in obtaining head pose and thus methods that explicitly rely upon the head pose may be unable to obtain a gaze estimate at all. An example of this issue is shown in Figure 2.8. Unfortunately, no calibration details were made available regarding the camera intrinsic parameters and the screen position relative to the camera.

The other state-of-the-art models that currently have the potential to meet the requirements set out are mostly based around the appearance-models that

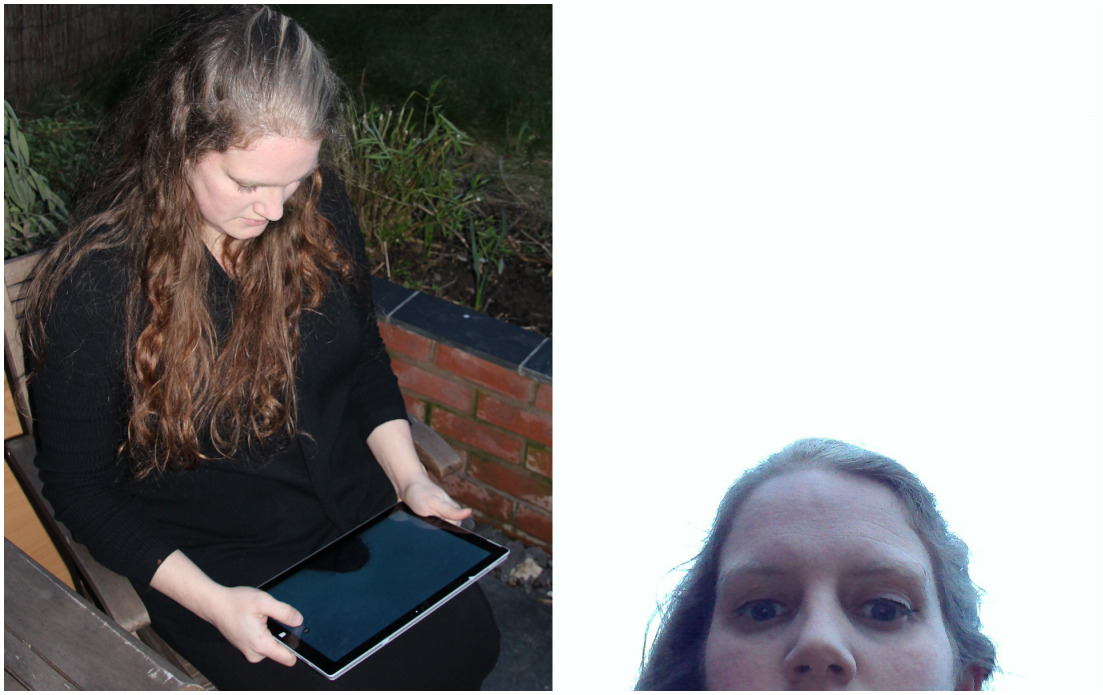


Fig. 2.8 It is frequently the case that when a user holds a tablet in a comfortable position that their head is only partially visible within the image of the on-board camera

have shown great resilience under conditions that are person and pose independent under cross-dataset conditions where no training would be required by the users. Such methods include a k-Nearest Neighbour (kNN) approach (Wood et al., 2016*b*), a Convolutional Neural Network (CNN) (Zhang et al., 2015), Random Forests (RF) (Sugano et al., 2014) and Adaptive Linear Regression (ALR) (Lu et al., 2014). These models have recently reported results on open source datasets such as EYEDIAP (Mora et al., 2014) and Columbia (Smith et al., 2013). In particular, the EYEDIAP dataset features both standard and high definition videos of a similar quality to those on modern mobile devices and while the cameras are staticly placed, the variation in head pose provides a suitable test-bed. The appearance based models achieve a Mean Angular Error (MAE) of between 10.5°(CNN) and 21.49°(kNN) (Wood et al., 2016*a*; Zhang et al., 2015). These seemingly large mean errors are symptomatic of the difficult assessment being asked of it, with large individual errors having a significant effect on the mean. Recently, a generative model-based approach was introduced that meets the requirements of pose and person independence (Wood et al., 2016*a*), achieving a MAE of 9.44° on the same data. Since it takes several seconds to estimate the pose for a single frame it can not be considered for real-time use on a current mobile device but sets the current highest benchmark through which new models can be compared.

2.6 Head pose tracking

Gaze tracking depends on both the head pose and the eye orientation relative to the head. While the literature shows there are many ways to determine a user's gaze, many of these methods make certain assumptions that are not valid for use in a 'mobile' case, or use technologies which would require additional modifications that are restrictive and unsuitable. Making any assumptions about either head-pose or eye-orientation will severely limit the effectiveness of any gaze tracker or constrain it to only limited scenarios. Furthermore, while additional components such as infra-red LEDs and stereo cameras may be helpful to obtain accurate information about the scene they should not necessarily be a requirement as they are not readily available on current mobile devices. Where there is currently an ideology to make gaze tracking more pervasive, they act as an

additional barrier to an input stream which is already considered restrictive. The goal then is to acquire the head-pose, i.e. the translation and rotation of the head, typically relative to the camera, in real-time. This process involves some element of detection and tracking within the 2D image in order to obtain a 3D output. Of course since the 2D image has effectively lost a dimension of data, it is vital that a realistic model of the head and camera are known to achieve a reliable estimate of real-world pose.

2.6.1 Measuring accuracy of head pose estimates

Due to the complexities of acquiring the ground truth data in real-world scenarios, measuring the accuracy of many applications within the computer vision domain is often very tricky. This is certainly true for the areas of head-pose and eye-gaze estimation. Since the head shape is not static it can be very difficult to define what its exact head pose is. Some works such as the Boston University (BU) (La Cascia et al., 2000) and Public University of Navarre (UPNA) (Ariz et al., 2016) head pose datasets acquire their estimate through the user wearing a tracking device (such as a *'flock-of-birds'* real-time tracker). This often takes the form of a head band firmly affixed over the head so as to not obscure the face. Of course there is no default position for it to be placed in, and so its output will be some (hopefully fixed) offset in translation and rotation relative to the real head pose. Additionally, it is difficult to ensure that its absolute translation values are measured to be correct and reliable. As such many researchers have taken to measuring the accuracy of a head pose methodology through the rotation parameters alone.

Other researchers have taken to using combined colour and depth (RGB-D) cameras such as the *Microsoft Kinect* or *Intel Realsense* to try to obtain the measurements without attaching devices to the participant. The information captured from these cameras tend to be noisy and since the incoming data is in raw pixel format, acquiring the output pose data has to be performed through another estimation process such as Random Forests (Fanelli et al., 2011; Tang et al., 2011) or Particle Swarm Optimisation (Padeleris et al., 2012). Thus the veracity of the 'ground truth' can be questioned.

Additionally, the depth and RGB components of the camera are offset some

distance from one another and therefore this needs to be taken into account. Most works again provide a rotation estimation relative to the first frame rather than any absolute translation and rotation estimates due to these problems (Padeleris et al., 2012). Alternatively a generic 3D head scan can be utilised to find an optimal fit to the incoming depth stream, with the pose parameters given relative to its default position (Fanelli et al., 2011). Funes-Mora and Odobez (2015) try to make a person specific template head model by using a 3D Morphable Model (3DMM) initialisation step in their EYEDIAP dataset (Mora et al., 2014), which also makes available the ground-truth gaze vectors.

These issues add up to significant difficulties in applying the head-pose to the gaze estimation process. Ideally we are looking to know the exact 3D locations of the user's eye centres in absolute terms. While this is extremely difficult with current measuring devices, it is important that the eye locations are determined through some fixed offset from the head pose origin. This origin needs to remain locally static (e.g. the average of the eye corners) in order to reliably estimate.

The measures produced are typically the Euler rotation angles of pitch, yaw and roll where the mean error for each, along with the standard deviation, can be acquired over all frames (Ariz et al., 2016). It is important to remember that these angle triplets are highly dependent on the order they are performed in and since there is no consistent way of performing these operations in the literature the Mean Angular Error (MAE) is reported. This value obtains the mean of all three rotation angles and allows models to be compared fairly. Frames of the image sequence where tracking is completely lost are usually removed as outliers. These frames can be a result of obfuscation or partially removing the head from the image. More likely, a rotation in pitch and yaw has occurred that is too far from its default 'front-on' position such that the main features of the face are no longer easily observable. State-of-the-art methods (Ariz et al., 2016) can achieve errors within $3 \pm 3^{\circ}$ using a generic shape model (from 720p RGB video at 30 *frames-per-second* (*fps*)), which is a realistic minimum target to build an eye-gaze estimation method upon.

2.6.2 Head tracking models

Since we aim at finding a mobile gaze-estimation solution, with the user able to move their head around, tracking the user's head within a series of images from a video stream becomes fundamental. There are two task to carry out to achieve this, firstly to *detect* the initial location within an image of the head and secondly to *track* that head through a series of images. The most successful and well-documented head detector in recent times is the Haar classifier from Viola and Jones (2001), which provides a good initial estimate of the head location. Once a head has been successfully detected, efficient tracking is performed by searching the local area of the current solution. This is usually sufficient provided the relative movement is not too great and the refresh rate is quick enough, i.e. frames processed per second is high enough. In recent years more accurate facial feature detectors have been designed to improve the accuracy of the detected model.

One of the most important issues when dealing with head tracking relates to choosing a representation of the head. We may choose to project a 3D shape into the camera image or perhaps more simply utilise a 2D shape model to track the head and subsequently estimate the 3D pose from the 2D shape (see section 2.6.4). Many of the representations involve capturing the texture from the image and capturing the relative movement over time. One of the simplest representations comprises a texture mapped cylinder (Cylindrical Head Model – CHM) (Xiao et al., 2003), which is formed by instantiating a cylinder into the camera scene and mapping the current image frame onto the cylinder surface. Since only the relative movement is determined, capturing displacement between the head shape and other items in the environment requires further calibration and additionally, like many tracking methods of this variety, the tracking tends to degenerate over time and requires re-instantiation regularly. Other simple shapes may approximate the head shape slightly better with a relatively small computational cost including ellipsoidal models (Choi and Kim, 2008) and sinusoidal models (Cheung and Peng, 2015).

Alternatively, models of the head shape and texture can be built beforehand, where we attempt to acquire a best-fit solution for the model to the new image. By approaching the problem with pre-learned knowledge about the face, we

can estimate relative distances between in-scene objects better, and also reduce cumulative tracking errors by always optimising the fitting to the learned data and using the previous tracking iteration as a guide only. Specifically, if we wish to track the shape and subsequently determine pose, we need to acquire more information about the distribution of the individual face components or features, such as the eye corners and bridge of the nose. Predetermined knowledge about a face shape often comes in the form of a Point Distribution Model (PDM) (Cootes et al., 2001). The PDM is capable of creating plausible shapes from a sequence of deformable points that are statistically learnt from a training set of marked up images of the shape. It is a simple linear parametric model in either 2D or 3D (see Figure 2.9), that given enough training data generalises well to even unseen data. Furthermore, by restricting the parameters to fall between plausible boundaries, the model can overcome issues with noisy data and occlusion.

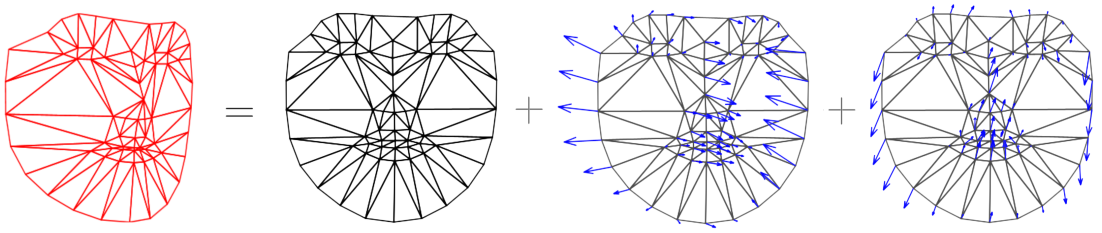


Fig. 2.9 A Point Distribution Model – we can create a large amount of different faces by simply shifting the mean face shape along its 2 most significant eigenvectors

The PDM is typically built by first aligning the collection of annotated training shapes. In 2D this is often achieved via a generalised Procrustes algorithm (Goodall, 1991) which normalises the shapes to a common reference shape (with common scale, translation and rotation). In 3D other solutions are often incorporated such as Iterative Closest Point (Rusinkiewicz and Levoy, 2001). When the shapes are aligned Principal Component Analysis (PCA) (Kirby and Sirovich, 1990) can be applied, which provides the mean shape along with a set of eigenvectors capturing the statistical variation of the shapes in the training set.

PDMs form the basis of many tracking techniques in the literature from simple Active Shape Models (ASMs) (Cootes et al., 2001) built from a relatively small amount of points, to 3D Morphable Models (3DMMs) (Blanz and Vetter, 1999) that are usually constructed via 3D range scanners and are therefore denser comprising of potentially thousands of points. Examples of these PDMs can be

seen in Figure 2.10. Finding the best distribution of these points within a given image is a difficult problem, due to the large variety of potential textual information that the image may provide. The ASM takes a straightforward approach in attempting to align the points with strong edges within the image and thus a predefined model of texture is not required. Numerous variants of the ASM have been observed in the literature including hierarchical and multi-resolution approaches allowing different parts of the shape to be modelled separately with varying levels of detail, which tends to provide faster training under a smaller set of training samples (Cerrolaza et al., 2012; Davatzikos et al., 2003).

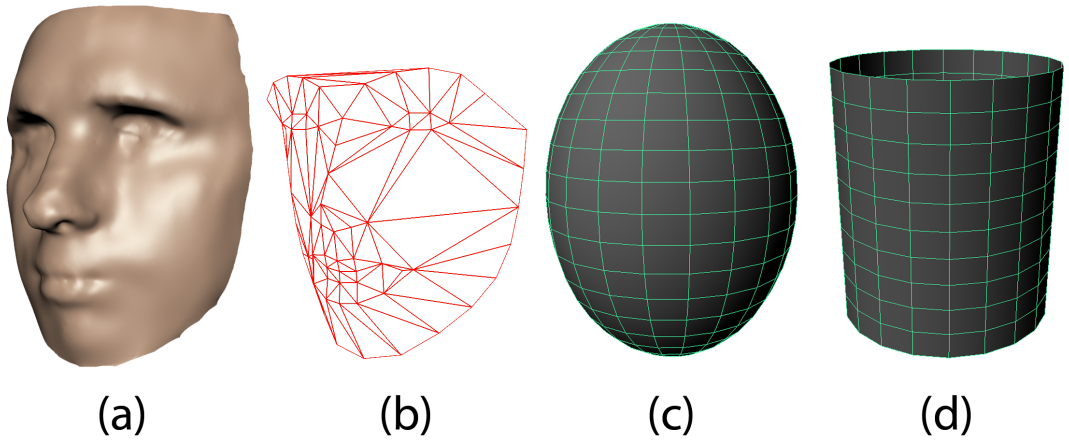


Fig. 2.10 Head models come in a variety of different complexities (a) a 3D Morphable Model (3DMM) captured with great accuracy from a depth scanner; (b) a 3D Point Distribution Model (PDM) which represents only a small sample of 3D points to approximate the face shape; (c) and (d) represent much simpler shapes which don't require training data and simplify calculations – here in ellipsoidal and cylindrical forms.

2.6.3 Generative models vs discriminative models

Having a predefined model of texture can be beneficial in constraining the search process in order to reduce false positives. The analysis of the image data can take two main forms: generative models, which are parameterised textures of the whole face (a holistic texture model) that when combined synthesise a new facial appearance, and discriminative models, which combine a number of local feature detectors (patch texture models) representing each point on the face.

By altering the parameters of a generative model, we can synthesise (generate) new faces and the search problem we are looking at comes down to determining which parameters most effectively describe a new texture from a given image. The discriminative approach comes down to a simple classification problem. For example, *does this new image patch represent an eye corner, or not?* By combining the response from all of the patches, we can optimise our shape parameters to best solve the problem.

The Active Appearance Model (AAM) (Cootes et al., 2001) is perhaps the most known and widely used example of a generative face model. The AAM uses databases of labelled pictures to build the PDM (shape model) and is combined with a texture model of the entire face. This texture model captures the statistical variation in the same way as the shape model via Principal Component Analysis. Together they form the definition of 'appearance', the parameters of which are minimised during the fitting process through a least squares gradient descent algorithm. This texture model depends on the individual pixels fed into the training set, of which there are many, and each pixel can have a wide range of values. Therefore the texture model is attempting to cover a very large subspace. One of the issues with such a model is that if the appearance of a new image does not match well with the training examples (i.e. if the person is not in the initial training set, or varying lighting conditions are not included in the training set) the fitting algorithm will struggle to minimise the appearance term. The original AAM also suffers from high computational complexity, since the Jacobian matrix which describes how the appearance and shape can change at any given time has to be computed for each frame. To achieve real-time performance Matthews and Baker (2004) describe the Inverse Compositional AAM which warps the current texture to a standard reference frame via a piecewise affine warp, which allows for the Jacobian (and Hessian) matrices to be precomputed.

The Constrained Local Model (CLM) (Cristinacce and Cootes, 2006) is a discriminative model as opposed to the generative model of the AAM. CLMs involve training small texture patches which correspond to parts of the face such as an eye-corner, or nose-bridge. Each patch searches its local area for a 'best-fit' and the pose update of the face is determined via a least squares approach from all the patch results. The result is further enhanced by ensuring that the face is still constrained by the learned shape-model which describes how a face can move.

Much of the literature investigating improvements to the CLM fitting process has been based on the type of patch used, and how best to amalgamate all the local response maps to globally optimise the solution. The format of the trained patch model can be of any classification tool as long as it produces a high response at the correct location. Typically a patch is trained on upright faces, and during fitting the image patch is rotated to match the previous known face position. The first CLMs trained their discriminative patches via a Support Vector Machine (SVM) (Wang et al., 2008). Other notable examples include correlation filters such as Minimum Output Sum of Squared Error (MOSSE) (Bolme et al., 2010) and Local Neural Fields (LNF), which learn non-linear and spacial relationships between the pixels (Baltrusaitis et al., 2013). These filters tend to produce more robust classifiers, although non-linear filters tend to be slightly more computationally intensive. A version of the LNF classifier is contained within the OpenFace toolkit (Baltrušaitis et al., 2016), which is freely available for comparison metrics.

The 2D response maps from all the patches need to be aggregated in some way to determine the optimal response. Determining the best combination of all pixel responses via an exhaustive search is too complex and therefore needs to be simplified. The simplest measure is defined as the Weighted Peak Response (Cootes et al., 2001), which simply assumes the coordinates of the patch with the highest response reflects the true (local) optimal position and to weight each patch's contribution on the final solution by their likelihood value (response value). To make better use of the entire response map, they can be simplified and replaced with simple parametric forms such as a Gaussian response (Paquet, 2009) or via Convex Quadratic Fitting (Wang et al., 2008). Later Saragih et al. (2009) proposed a non-parametric form solution called Subspace Constrained Mean-Shift that represents more robustly the patch responses.

2.6.4 Obtaining the 3D head pose

One simple way to extract an estimated 3D pose from the 2D PDMs is through the Pose from Orthography and Scaling with ITerations (POSIT) algorithm (Dementhon and Davis, 1995), which estimates pose via a given 3D rigid model and 2D point correspondences. Since it is based on an orthographic model with scaling (a weak-perspective view frustrum), the full camera perspective is not taken

into account. Xiao et al. (2004) attempt to jointly constrain the shape in 2D and 3D via a 'combined 2D + 3D AAM'. The 2D shape generated from the model is further constrained to satisfy the limit of a 3D PDM. The solution described works with a weak-perspective model on the principle that any 3D shape can be represented in 2D by adding 6 additional parameters and a balancing weight. Baltrušaitis et al. (2012) describe the CLM-Z which matches a 3D PDM to 3D depth data coming from a RGB-D camera while Martins et al. (2012) simplify the optimisation process by proposing a 2.5D AAM that works with a 3D PDM and 2D texture model under a full perspective camera. The 3D PDM decreases the number of eigenvectors needed to statistically represent the face (since a 3D shape remains relatively constant unlike a 2D representation of a 3D shape). Additionally, the full perspective model better represents the likely shapes seen from cameras positioned reasonably close to the head. However, since AAMs are generative methods they typically require texture information from the specific user to perform with high accuracy which means they won't generalise as well to the average user or even different lighting conditions. The user would likely be required to annotate parts of their face and build a new model.

2.7 Summary

From the literature we see that the development and use of gaze methodologies often requires simplifications or approximations in either the head pose, eye orientation or both. To meet the requirements of the research questions on mobile devices, many of these simplifications do not hold true and thus the focus needs to be on devising methodologies which are able to incorporate both with a suitable level of detail and accuracy. Of course, one of the major issues is performing these calculations in real-time from noisy data.

Head pose estimation is often considered an aside from head tracking in an image, however they are separate problems with subtle differences. Head pose estimation requires 3D output and thus additional processes are needed on the resultant 2D head tracking estimate. Since there is no guarantee that the 2D PDM can generate a viable 3D face shape, it is perhaps better to further constrain the problem by estimating the 6 head-pose degrees of freedom directly via so called 2.5D models, which utilise a 3D PDM within the 2D image.

Of the two gaze tracking paradigms, we see that the interpolation-based methods can be easy to calculate but often too restrictive (in head pose, and the requirement for recalibration), whereas the best geometric methods have shown that they can provide good accuracy when the parameters of the system are calibrated and setup correctly. Obtaining these parameters accurately can be difficult and time-consuming, which is why these systems are less commonly seen in commercial systems and why the literature tends to constrain certain elements of their solutions (e.g. by using chin rests to prevent excessive head movement). Nevertheless, the geometric methodology has the potential to be well suited to the case of gaze tracking on mobile devices, particularly if synchronised with the head pose model.

The next chapter introduces the research tools and methods that will be used to experimentally evaluate the proposed solution.

Part II

Research Methodology

Chapter 3

Research Tools & Methodology

In this chapter the tools and datasets that are used through the remainder of the thesis are discussed. It is crucial that any proposed methods are rigorously tested on publicly available datasets so that they can be evaluated and tested against other similar works seen in the literature now and in the future. Additionally, since datasets recorded on mobile devices containing all the required calibration parameters do not exist, a small test dataset was recorded for this purpose and its construction shall be discussed here. The tools and datasets that can be used to train models are also to be discussed, as well as the state-of-the-art methods seen in the literature that will be used for comparison during testing.

3.1 Training data

Due to the difficulty in obtaining a large number of hand-annotated images, research institutions have funded the creation of large datasets which have been shared with the research community. The MultiPIE dataset was established by Carnegie Mellon University for research purposes (Gross et al., 2010). It is a large annotated image collection carried out over numerous sessions and captures a large number of people from synchronised cameras set at 15° intervals around the *y-axis* (see Figures 3.1 and 3.2).

It has around 750,000 training images in total with 337 people captured under 19 different illumination conditions (Figure 3.3). Each image also has an annota-

Fig. 3.1 Multi-PIE dataset captures data simultaneously from many different angles

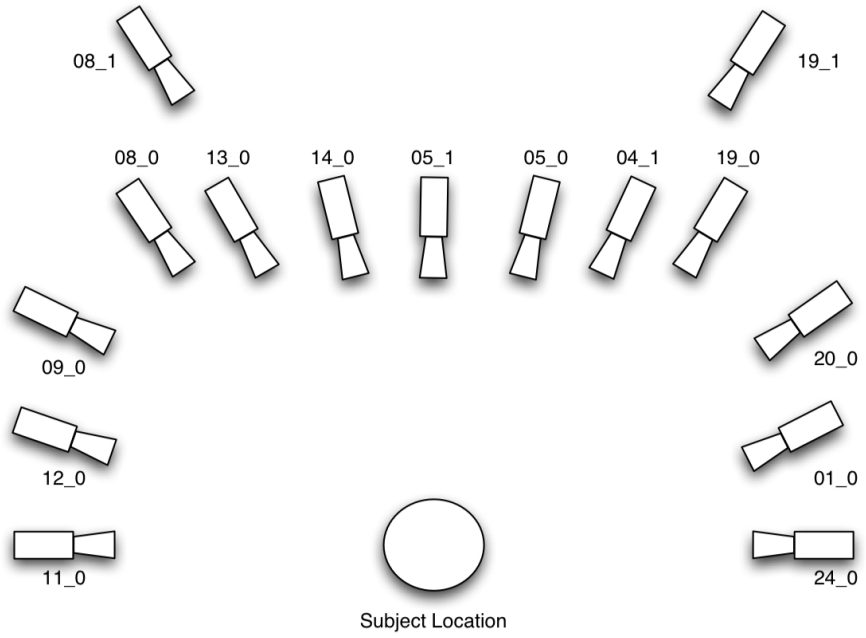


Fig. 3.2 Multi-PIE dataset camera locations

tion file that marks the image of 68 predefined points on the face that can be used for training purposes. The participants were also asked to perform a number of different facial expressions to capture deformation of the face structure (Figure 3.4).

Acquiring a large amount of data from the eye region is very difficult and no substantial dataset of tagged data currently exists. For this reason synthetic eye models are appearing in the literature that are freely available for the research community to use. One such program is UnityEyes by Wood et al. (2016*b*) where

Fig. 3.3 Multi-PIE dataset illumination variation

Fig. 3.4 Multi-PIE dataset expression variation

some examples of the generated eye can be seen in Figure (3.5). The program models the complex eye region and materials from raw depth scans and generates anatomically inspired procedural geometry for the eyelids to create lifelike high resolution images. UnityEyes provides controls to orient the eyeball specifically so that the direction of gaze can be changed and automatically tags numerous locations on the eye that can be used for training purposes.

Fig. 3.5 Numerous outputs from the UnityEyes eye-region generator. The program models changes in face-shape, colour, lighting and eye-rotation, from which models of texture and shape can be built.

3.2 Test datasets

Several datasets are available for the testing of head-pose and eye-gaze methods including one constructed specifically for this work. Table (3.1) provides a quick reference to what they provide. For the comparison of head pose algorithms a dataset was recently established by Ariz et al. (2016). The dataset was created by the Gaze Interaction For Everybody (GI4E) group at Public University of Navarre (UPNA) with a belief similar to this work – that the head pose has a strong influence on eye-gaze and therefore a suitable test ground for facial landmarking and pose is required. Each individual in the dataset had a *'flock-of-birds'* 3D Guidance TrakSTAR firmly fitted to the top of their head. The device and user were carefully calibrated to ensure correct rotation and translation from the camera. A 3D point model of the user was also acquired through a novel tagging method, allowing for the offset between the tracker and user to be calibrated. Previously, a common dataset for pose estimation was the BU head tracking dataset (La Cascia et al., 2000) however the dataset is nearly two decades old and features very low camera resolutions by modern standards (320 × 240), limiting its usefulness for eye-gaze tasks.

HEAD POSE DATASETS			
Dataset	UPNA (GI4E)	EYEDIAP	DMUHAG
Participants	10	16	11
Videos per Participant	12	4 VGA 4 HD (Synchronous)	2
Frames Per Video	300 (30FPS)	4500 (30FPS)	1800 (30FPS)
Camera Resolution	1280x720	640x480 (VGA) 1920x1080(HD)	1920x1080
Average Head Image Size	200x220	70x120 (VGA) 320x300(HD)	400x500
Type of movement	Free/Guided	Free/Stable, Free/Large Motions	Free
Ground Truth	*'Flock of birds'* 3D Guidance TrakSTAR	RGB-D *Microsoft Kinect*	RGB-D *Intel Realsense F200*

EYE GAZE DATASETS			
Dataset	Columbia	EYEDIAP	DMUHAG
Participants	56	16	11
Videos per Participant	105 (Discrete Images)	4 VGA 4 HD (Synchronous)	2
Head poses	5 (Chin-rest)	Continuous	Continuous
Eye gaze targets	21	Continuous	13
Target Type	Discrete	Discrete/ Smooth pursuit	Discrete
Camera Resolution	5184x3456 (Calibration Estimated)	640x480 (VGA) 1920x1080(HD)	1920x1080
Average (Single) Eye Image Size	320x120	23x11 (VGA) 70x50(HD)	96x36

Table 3.1 Test dataset details

Another recently proposed dataset is the EYEDIAP dataset that provides two different video streams which challenge the models in different ways (Mora et al., 2014). First, a very low resolution video captured of the head and eye from a RGB-D camera (*Microsoft Kinect*), and secondly, higher definition videos but with all images captured from the side (approximately 15° to 40°). The streams were synchronised via 5 LEDs within the image frame of each. The dataset was mainly developed to isolate conditions for eye-gaze tracking, one of which of course is the head pose. The videos have a number of differing specifications including a stable head position along with videos where participants were asked to significantly move their head. During all videos the participants had to look at targets either on a 24 inch monitor screen or a floating target that moved in 3D space. The screen targets come in two varieties – discrete targets and continuous targets which follow randomly generated Bézier curves to evaluate smooth pursuit. The ground-truth head pose was acquired by fitting a 3D mesh of each user to the video stream as described in Mora and Odobez (2012).

For a more controlled test of eye-gaze, the Columbia image dataset (Smith et al., 2013) is available with very high resolution images (5184 × 3456) taken with a telescopic lens. The participants head pose is controlled through the use of a chin-rest. Five different head pose angles (−30° to +30°) are acquired (asynchronously). From each camera angle, 21 targets aligned in a grid (7 × 3 in ±5° horizontal and ±10° vertical increments) are observed on a wall approximately $2.5m$ away. There are 56 people in the dataset which cover a range of different ethnic backgrounds and also a mix of gender and age, giving opportunity to test an algorithms generalisability.

3.2.1 DMUHAG dataset

To test specifically on a mobile tablet device, a small dataset has been created specifically for this work – the De Montfort University Head And Gaze (DMUHAG) dataset. The image data is captured by an *Intel Realsense F200* camera which simultaneously captures 1080p RGB footage along with a depth stream. Similar to the EYEDIAP dataset, the depth stream is utilised to obtain the ground-truth of the head pose by first capturing a 3D model of each participant's head and then estimating the pose within each frame using a simple

Particle Swarm Optimisation (PSO) process. A particle in this case represents an estimate of the 6 degrees of freedom for the current frame and a fitness value determining how well it matches the incoming depth frame data. The depth frame data is itself mapped to 3D Cartesian coordinates to determine the actual intersection points.

Fig. 3.6 Sample of a captured 3D head model from multiple angles

In order to determine the fitness of a particle the incoming depth frame needs to be compared with the expected depth at the estimated pose, and so a suitable ground-truth representation of the head is required. To achieve this, first the user's head was scanned using the *Intel Realsense* SDK supplied with the camera (Figure 3.6). The correct placement of the head was ensured by manually annotating points on the captured model within 3D modelling software. A compact representation of depth was then achieved by forming a plane with its centre located at the head origin $\mathbf{O}$ and projecting the 3D points of the face onto it. Letting the plane have unit vectors $\mathbf{i}$ and $\mathbf{j}$ along it, and a unit normal $\mathbf{n}$, the 3D points $\mathbf{x}$ of the face can be sampled and projected onto the plane that when bounded forms a 2D square pixel image as

$$\begin{aligned} u_k &= (\mathbf{O} - \mathbf{x}_k) \bullet \mathbf{i} \\ v_k &= (\mathbf{O} - \mathbf{x}_k) \bullet \mathbf{j} \\ d_k &= (\mathbf{O} - \mathbf{x}_k) \bullet \mathbf{n} \end{aligned} \tag{3.1}$$

where u_k and v_k form the horizontal and vertical coordinates on the square for the k^{th} sampled point and d_k represents the perpendicular distance away from the face. An example can be seen in Figure (3.7), with the lighter areas representing greater distances.

On every frame, 100 particles are instantiated around the last known pose location, each representing a slight random pose adjustment along one or more of the 6 degrees of freedom. By projecting the incoming depth information onto a square located at the particle's pose, the fitness of the particle can be acquired by directly comparing the individual pixels with the corresponding pixels on the ground truth image and taking the summed squared difference (where both images have valid depth information). Iteratively, the particles are shifted towards the best fitting particle and re-evaluated until convergence or a maximum number of iterations has been reached.

To record the data the camera is firmly attached to the top of a *Microsoft Surface Pro 3* tablet computer and data is captured of 11 participants in an uncontrolled but reasonably well-lit environment. Ethical approval was gained

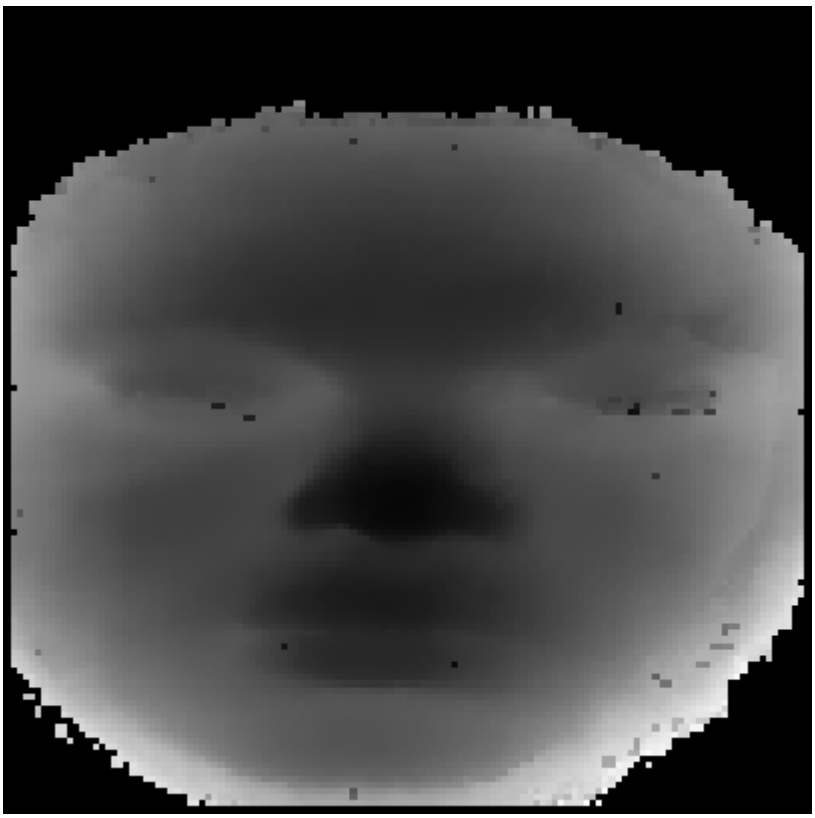


Fig. 3.7 A ground-truth sample depth, represented as a square pixel region. Each pixel represents the desired perpendicular distance away from the face, with the lighter areas representing greater distances. Pixels that are pitch black here do not have a valid depth measurement and are not included in the particle fitness estimates.

and all participants had the right to withdraw at any time. Additionally, all participants were informed of the purpose of the data collection and agreed to have their head scanned and video data collected and stored securely. There are 3 women and 8 men in the dataset with varying ethnic backgrounds and 3 participants were wearing glasses at the time of the recordings. During the data collection users had free head movement and were asked to follow a target as it moved to 9 different positions on the tablet screen (Figure 3.8). The users were asked to look at an additional 4 points to the left, right, above and below the screen as these points are often used as modalities of gaze input (Istance et al., 2009). Additionally they also enforce a larger range of head movement to test upon. Each captured video was approximately 60 seconds long.

Two varieties of videos were captured specifically to test the effect on tracking when the tablet is held in the user's hands – *static* and *dynamic*. In the *dynamic* case, the tablet is held in the hands and the captured footage is typically far less stable. Also perhaps due to the weight of such devices, they are often supported

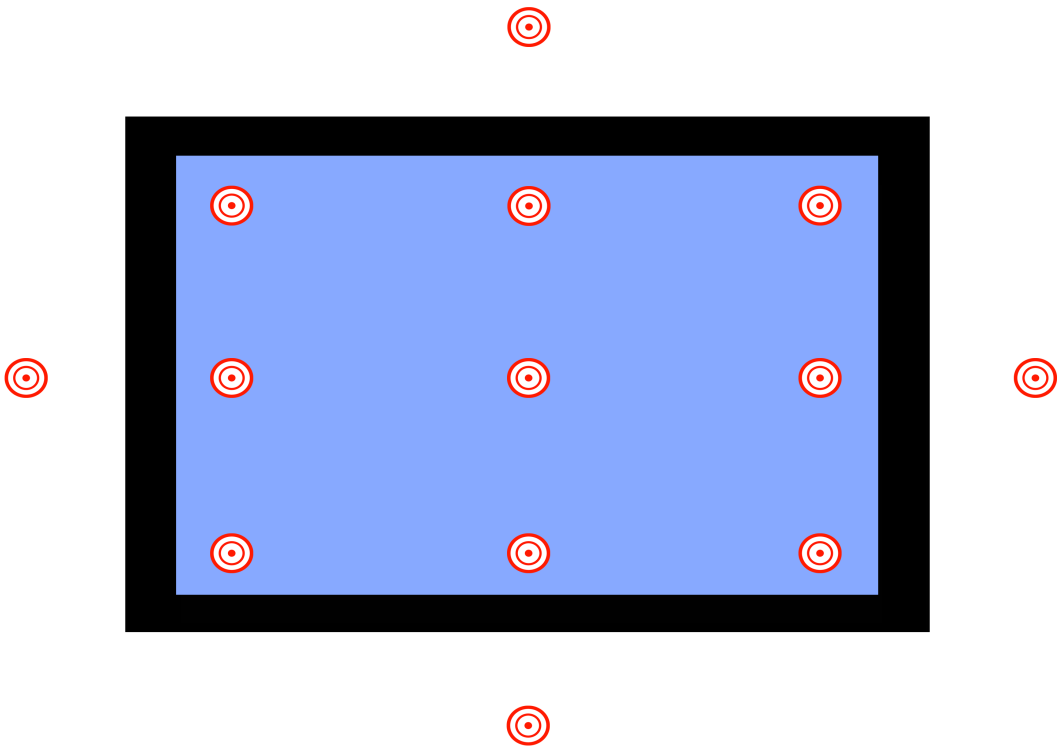


Fig. 3.8 The 13 target points around the tablet screen in the DMUHAG dataset.

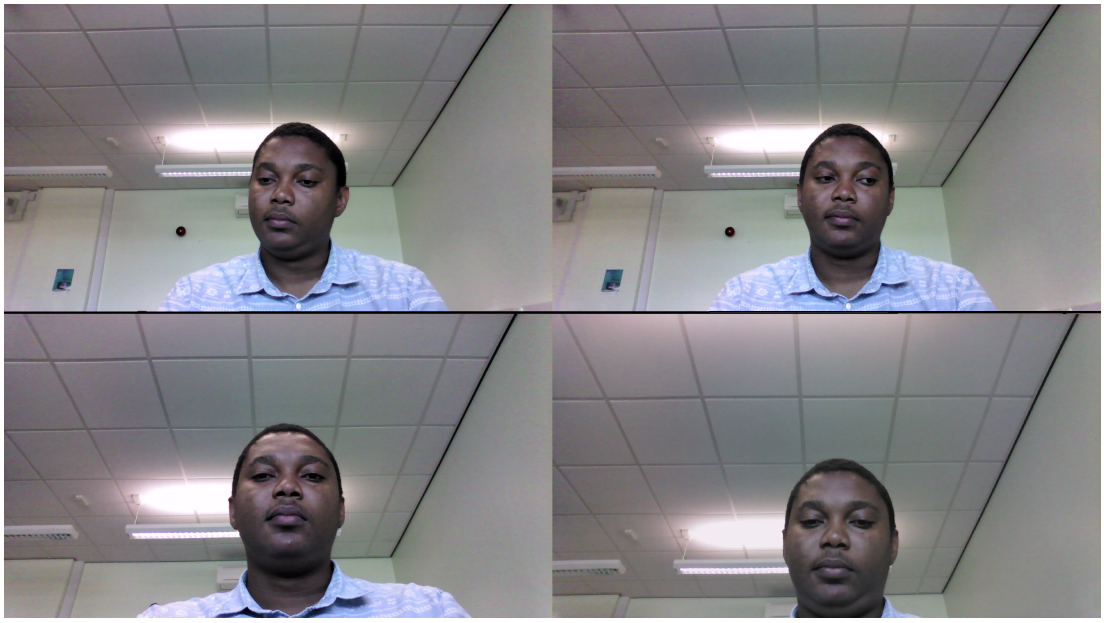


Fig. 3.9 Top row – *static* conditions with the tablet placed on a desk. Bottom row – *dynamic* conditions with the tablet held in the hands. Holding a mobile device in the hands can cause additional complexities from camera shaking and frequent environment changes, most commonly through tilting the device toward the ceiling

by being held in the lap or otherwise lower down than the chest, which provides an upward view of the face. It also can have the effect of limiting the gaze vector to only downward trajectories that often restrict the visibility of the iris due to a lowering of the eyelids. A *static* control set of videos were captured with the tablet device placed on a desk in front of the user. An example of the *dynamic* and *static* cases can be seen in Figure (3.9).

The camera was calibrated as standard using the open source OpenCV library (Bradski, 2000), with 100 image samples being taken of a checkerboard pattern from different perspectives. Additionally careful consideration was taken to calibrate the monitor position using the same tools with a method similar to Takahashi et al. (2012). As shown in Figure (3.10) circles were placed on the tablet display and were observed by the RGB camera through a mirror from many different angles. In practice it can be difficult to capture enough angles for a good estimation and mirror distortion also plays a part. To simplify the process and improve the accuracy in this work the depth stream from the camera was used to determine the mirror plane by sticking masking tape to its surface. A number

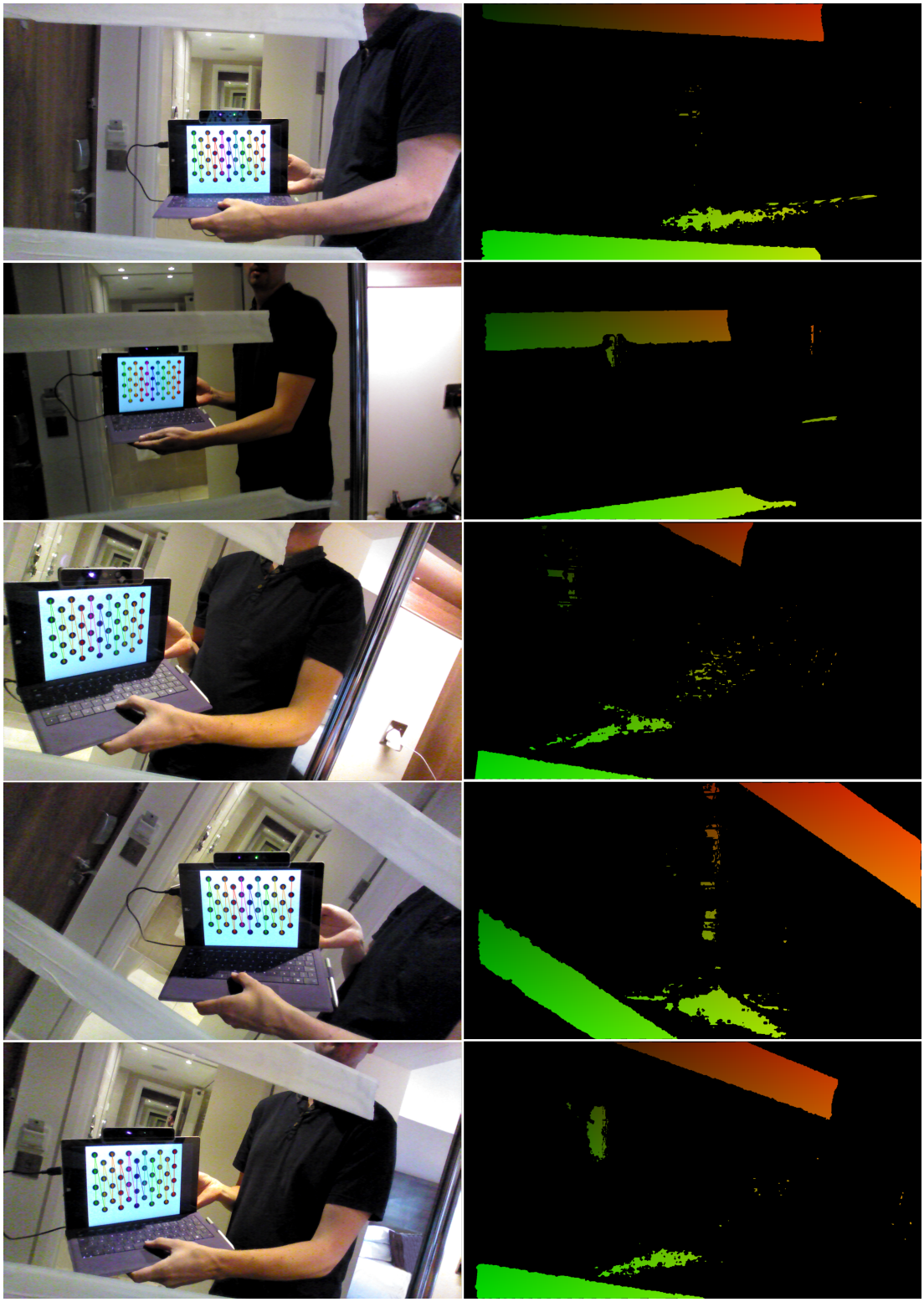


Fig. 3.10 Depth camera calibration

of 3D points on the surface can then be sampled to estimate the plane through a least squares approximation. Accurately determining this plane greatly reduces the number of samples required (12 in this work) and removes the main source of error from the algorithm.

3.3 Research methodology

Suitably testing a method to estimate gaze from a video stream on a mobile device can be difficult due to the nature of the incoming data, with millions of pixels each frame under unconstrained conditions. In this work 3 important groups of factors have been isolated – the *Camera*, *Environment* and *User*.

Within the *Camera* factors, the resolution of the camera is perhaps the most important as this determines how much pixel-data is available within the image as a whole. It should be clear however that this cannot be considered in isolation because what needs to be determined is exactly how much of this pixel data actually represents the face and eyes. An example of this can be seen in Figure (3.11). The shorter the focal length, the wider the angle of view captured by the perspective camera and vice-versa. Additionally, if the user simply moves away from the camera, the available pixels of the head is decreased. It is better then when talking of resolution, to clarify it in terms of detail of the region by the average pixel areas available.

The other important *Camera* parameter to think about is the *frames-per-second* (*fps*) of the captured video. A higher number of frames captured per second allows for smoother transitions between the frames and therefore it is likely that this would lead to less occasions when the tracking fails. On the other hand, processing the data at higher frame rates may not be currently possible due to the limited computation power available. While all the test datasets have videos that run at 30*fps*, more important is how fast algorithms can run in real-time on a mobile device. The average computing time and *fps* will be recorded on a tablet device where ideally any proposed models will match or exceed the the possible capture rate and this will be commented on within the results. The test sets also have cameras with a variety of resolutions ranging from a very high professional camera with a telescopic lens to low quality standard definition (SD)

Fig. 3.11 A decrease in resolution limits the available data around the eye region to detect eye features. The image resolution was twice down-sampled in professional imaging software to simulate the effect.

video feeds.

The second group of factors to consider are those relating to the *Environment*. It is common for many computer vision operations to perform poorly under complex lighting conditions and detailed backgrounds, and such factors are difficult to manage for a controlled experiment without having a large volume of video test data. For this work a more important consideration is how holding the mobile device in the hands compares to the case where the mobile device is static and placed on a table. When holding the device the camera and monitor move synchronously, commonly in a jerky manner which can cause large changes in pose and also image artefacts. The DMUHAG dataset has been create specifically to determine how much this factor influences the robustness and accuracy of the model.

The final group involves the *User* variability. Since every user's face and

eyes are different, video data should be acquired that captures as many different characteristics as possible including gender, ethnicity, skin and hair colour, as well as more subtle variations in shape of the nose, mouth and thickness of eyebrows. The proposed test and training sets cover a wide variety of people to capture as much of this variability as possible. Of particular interest are those users that have parts of their face occluded. This includes those that have beards, longer hair, or wear glasses that cover and distort the eye region such as in Figure (3.12). All of the datasets include people with some of these features and therefore careful consideration can be given as to how they impact on the results.

Fig. 3.12 A user with glasses which cause a large distortion effect around the eye region

3.3.1 Comparison models

As the Columbia, EYEDIAP and UPNA datasets are publicly available, there are a number of models in the the literature that have been tested on them. For head tracking, state-of-the-art methods such as the Active Shape Model (ASM) (Cootes et al., 2001) and Active Appearance Model (AAM) (Cootes et al., 2001) have been available for a number of years. A vast number of varieties have been documented over the years, but for generality and fair comparison it is important that the training and testing data does not overlap. While ASM and AAM are

very common, they are typically only used for 2D feature tracking. Head-pose is often an afterthought and frequently methods such as the POSIT algorithm (Dementhon and Davis, 1995) or similar are subsequently applied to the result. They can therefore be considered to be a two-stage process with the 2D features first being optimised followed by the fitting of a static 3D model to the points through an image projection method. The choice of 3D model is as important as the underlying method itself and fortunately the authors of the UPNA dataset have tested an ASM and AAM with a number of head models including a cylinder, a generic head model and also the real 3D points of the participant (Ariz et al., 2016). Each have shown impressive results with the data for roll, pitch and yaw angles individually published alongside the Mean Angular Error (MAE) in degrees.

For eye-gaze a recent trend seen in the literature is the success of appearance-based models that can work cross-dataset and be used by a variety of people without calibration. Examples of such models are a k-Nearest-Neighbour (kNN) approach (Wood et al., 2016*b*), a Convolutional Neural Network (CNN) (Zhang et al., 2015), Random Forests (RF) (Sugano et al., 2014), Adaptive Linear Regression (ALR) (Lu et al., 2014). These models have reported strong results (Wood et al., 2016*a*; Zhang et al., 2015) and the newly proposed model will be compared against them where data is available. The suggestion was made that model and shape-based methods can not compete with the generality of these methods on low quality images and this claim will be assessed. For EYEDIAP, Funes-Mora and Odobez (2015) utilise the underlying 3D data from the depth camera to estimate head-pose and eye-gaze and so these results will not be compared. A recently proposed generative 3D Morphable Model (3DMM) (Wood et al., 2016*a*) was also tested on these datasets with state-of-the-art results and although it does not run in real-time can be used as a high watermark. Additionally a shape-based method (Wood and Bulling, 2014) used on tablets will be assessed although it has only been shown to work with the user at a distance of around $20cm$ away due to the high volume of image data around the eye being required to effectively isolate the iris. The errors themselves are typically expressed as the mean or median angular error between the real and estimated 3D gaze vectors.

3.4 Summary

This chapter presented the training data for both the head and the eyes. In particular, the synthetic eye model provides access to data that has been previously extremely difficult to acquire, although of course any simulated model is likely to have some drawbacks and these will be kept in mind throughout the thesis. Both provide a large number of samples through which new models will be established in the following chapters.

The criteria for testing has also been discussed with a focus on the *Camera*, *Environment* and *User*. A number of test sets have been discussed that will allow for their evaluation as well as numerous works seen in the literature which will allow for comparison to the current state-of-the-art. In particular a small mobile dataset was introduced in DMUHAG that can be used to evaluate how well a new model adapts when the tablet device is held in the hands of the user. The next part of the thesis will introduce a new model specifically designed to try and meet these criteria.

Part III

Proposed Solution

Chapter 4

3D Head Pose Estimation for Mobile Eye Gaze

From the literature analysed in Chapter 2 it is seen that in order to gain a realistic chance of achieving a usable eye gaze solution on mobile devices we need to take into account the head pose. As aforementioned, the required head pose method needs to be robust, fast and unobtrusive through any calibration process and during use. This chapter will detail a new model composed to meet these specific criteria.

4.1 2.5D Constrained Local Model

To track the head efficiently with a stream of video data, a novel approach utilising 3D shape models with 2D texture information is needed. The model is named the 2.5D Constrained Local Model and emphasises the unique combination of 3D and 2D information. This work extends previously published work that investigated the potential for this kind of tracker in determining the gaze from a mobile device (Ackland et al., 2014). The work showed that the eye corners could be identified robustly on an unmodified tablet device and led to the development of the final model detailed in this chapter and a model for eye-gaze built on top of it that will be explored in Chapter 5. The model takes inspiration from the 2.5D Active Appearance Model (Martins et al., 2012) but deviates in a number of key areas

such as in the use of a discriminative model that uses small texture patches around important areas of the face rather than a generative texture model. As such, the model contains a 3D PDM which contains knowledge about the shape of the face, and 2D texture patches around small sub-regions of the face which replace the holistic texture. The texture patches are built so as to work with as many people as possible by utilising robust filters that efficiently work in the Fourier domain and have been shown to have high discriminative properties under varying environmental conditions.

The approach builds the shape model directly in 3D with a full-perspective camera model, which firstly constrains the number of possible face combinations in the 2D image (creating a lower dimensional problem space) and secondly, by definition, provides the 3D head pose. The full-perspective camera allows us to 'project' the 3D PDM into the camera image space allowing for real-world distance estimation. The model can also correctly accommodate radial and tangential distortion that is often prevalent on low-cost cameras such as webcams.

This thesis expands upon the earlier work in several important ways. Firstly, it takes a 'Multiview' approach to the problem to overcome one of its major weaknesses - the change in texture information as the model rotates and translates. The 2D texture information, by construction, cannot take into account the variety of possible textual differences even from a single user, often limiting its usefulness and restricting the user to predefined training conditions such as remaining front facing to the camera at all times. This problem is tackled by training several sets of image patches taken from training data capturing users at a variety of different angles, and dynamically switching between them at runtime based on the current estimation of the user's head pose. Secondly, by utilising the perspective camera model a Jacobian matrix is analytically derived which describes how changes in the parameters of the model create changes in the point positions within the image. Additionally, the PDM is created in a unique way such that the pose is always defined with its origin between the eye corners, helping the head-pose model synchronise with the eye-gaze model discussed in Chapter 5 while also lowering the computational complexity by reducing the variability within the model. Finally, a global face alignment method is introduced which improves the robustness of the model during fast head movements and/or low camera frame-rates. The cropped image of the head is tracked via a dynamic

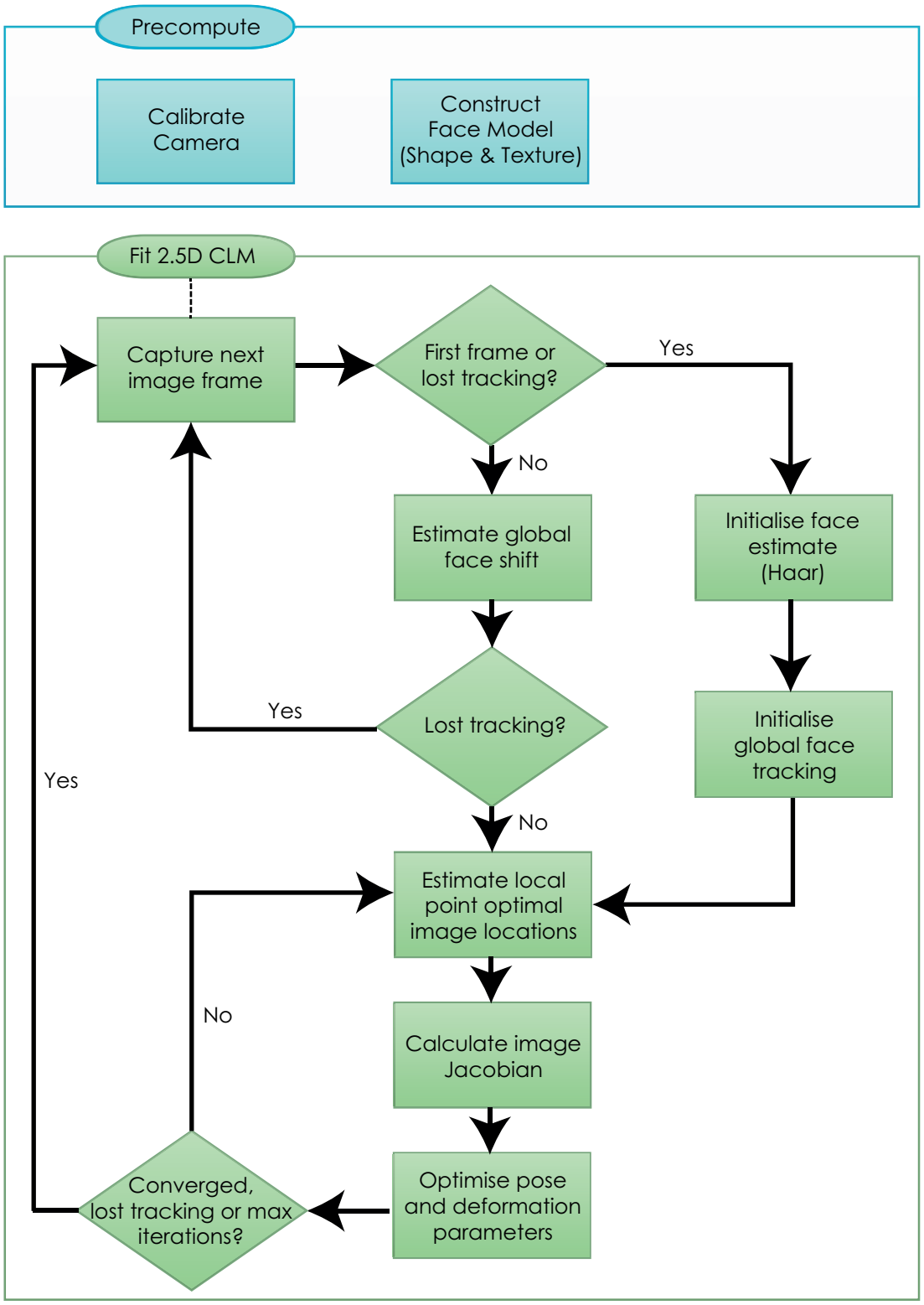


Fig. 4.1 2.5D Constrained Local Model flowchart

MOSSE filter which updates over time and therefore adjusts to changes in environment and lighting. It also provides an ability to determine a loss of tracking which the smaller patches fail to provide. The general flow of the algorithm can be seen in Figure (4.1).

For clarity, this chapter covers the following main novel contributions:

- 2.5D Constrained Local Model for *direct estimation* of the pose parameters
- Optimised Multiview MOSSE filters
- Perspective camera Jacobian matrix
- Shape model construction *for* eye-gaze synchronisation.

4.2 What do we mean by '3D head pose'?

The 'pose' of an object defines some measure of both translation and rotation. In order to measure the pose in absolute terms, it is imperative that any trained models are built using 'real-world' measurements. This implies that by having a correctly measured 3D head model, we can accurately track its translation and rotation from any camera (provided we have calibrated and obtained the cameras intrinsic values). Furthermore, this means that unlike other tracking algorithms which are effectively only tracking relative movements, no scalar or balancing weight parameters are required.

Throughout this thesis, the 'world' coordinates are defined from the point-of-view of the camera, that is to say, the camera acts as the origin with the camera always pointing down the positive *z-axis*. From the cameras perspective, the positive *y-axis* points downwards and the *x-axis* points to the right creating a right-handed coordinate system as shown in Figure (4.2). Of course under this definition the camera pose remains fixed at all times and any 'mobile' interactions involve the motion of all other objects in the world moving relative to it. The presented image frame is dependent on the focal length f in pixels between the camera pinhole point and the image plane. If the pixels are non-square, perhaps due to a small camera defect, the focal length can be defined in terms of its horizontal and vertical components f_x and f_y, which are approximately equal.

The camera has a 'principal axis' that passes through the camera pinhole and is perpendicular to the image plane at the 'principal point' (c_x, c_y), often at or near the centre of the image.

4.2.1 The pose parameters

The pose $\mathbf{P}$ of the head can be described at any time via a 3x4 matrix, with 9 parameters forming a traditional 3D rotation matrix $\mathbf{R}$ (using Euler angles) and 3 parameters representing the translation values $\mathbf{t} = [t_x, t_y, t_z]^T$ along the *x, y* and *z axes* measured in millimetres (*mm*).

$$\mathbf{P} = \left[\begin{array}{ccc|c} R_{00} & R_{01} & R_{02} & t_x \\ R_{10} & R_{11} & R_{12} & t_y \\ R_{20} & R_{21} & R_{22} & t_z \end{array}\right] = \left[\mathbf{R} \,\middle|\, \mathbf{t}\right] \tag{4.1}$$

To simplify our problem, we recognise that there are in fact only 6 degrees of freedom (three dimensions in rotation and translation) so formulate our problem in these terms using the Rodrigues rotation formula. This formula defines the rotation as an 'Axis-Angle' relationship where the rotation is defined as an angle θ around an arbitrary unit axis $\boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]$ in the following way:

$$\mathbf{R} = \mathbf{I} + sin\theta\ \boldsymbol{\Omega} + (1 - cos\theta)\ \boldsymbol{\Omega}^2 \tag{4.2}$$

where $\boldsymbol{\Omega}$ is the cross-product matrix for the vector $\boldsymbol{\omega}$:

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{4.3}$$

The three rotation parameters can be efficiently stored in a vector $\mathbf{r} = \theta\boldsymbol{\omega}$

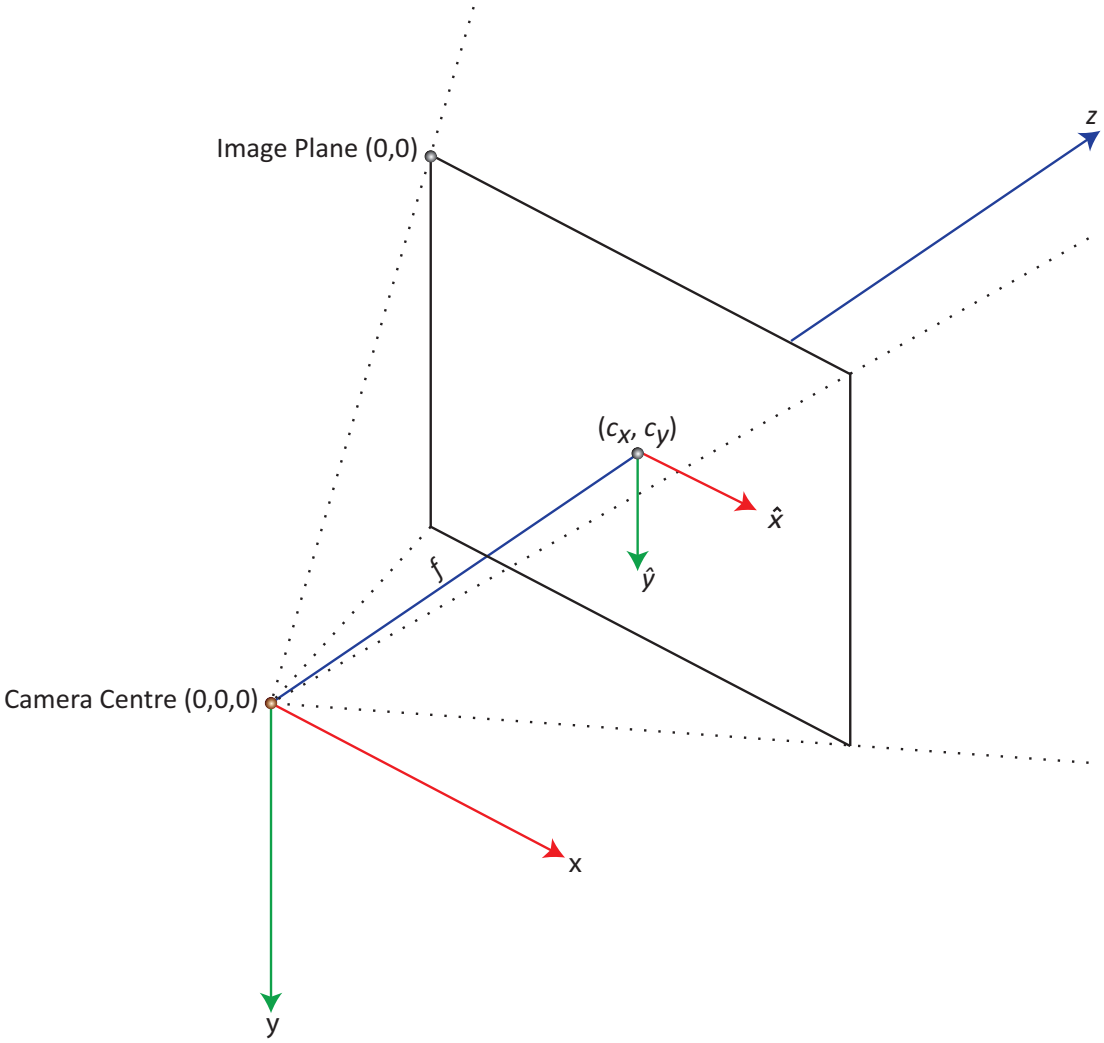


Fig. 4.2 The persepective camera model

where

$$\boldsymbol{\omega} = \frac{\mathbf{r}}{\theta}, \qquad \theta = \sqrt{r_x^2 + r_y^2 + r_z^2} \tag{4.4}$$

4.2.2 The Perspective Camera Model

Different webcams have a wide spread of varying intrinsic camera parameters (such as focal length) under which our head tracking model would produce vastly differing and inaccurate estimates. To capture as much of this variability as possible a full-perspective camera model is used to project the information from

3D to 2D.

A 2D point in the image $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ can actually represent an infinite number of potential 3D points. This relationship can be captured through the use of homogeneous coordinates where 3 values $[x^H : y^H : w^H]$ are used to represent the 2D Euclidean coordinate. The function E used to convert homogeneous coordinates to Euclidean coordinates is defined as

$$E\left(\begin{bmatrix} x^H \\ y^H \\ w^H \end{bmatrix}\right) = \begin{bmatrix} \frac{x^H}{w^H} \\ \frac{y^H}{w^H} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{y}} \end{bmatrix} \tag{4.5}$$

The camera intrinsic parameters play a key role in determining the homogeneous coordinates of a particular 3D point $\mathbf{X}$ observed in the world from the camera location. These parameters can be represented by a matrix $\mathbf{K}$

Fig. 4.3 Calibrating the camera. The squares are all planar and have sides exactly 25mm long. Accurately estimating their corners within the image allows for estimation of the camera parameters, with more precision gained when taking a large number of samples from many different angles.

$$\mathbf{K} = \begin{bmatrix} f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

where f_x, f_y represent the camera focal length; c_x, c_y represent the camera principal point and α the skew parameter. These values can be determined via a standard calibration procedure where a checkerboard pattern with known dimensions is observed multiple times within the image frame (see Figure 4.3).

In addition, real camera lenses are likely to have some minor tangential or radial distortion. An extreme example of this would be a fisheye lens which causes extreme distortion around the edges of the image. Provided the distortion values are known through some calibration procedure they can be easily supplemented into the camera projection model without loss of generality. A discussion about how to do this alongside expected projection errors can be seen in Weng et al. (1992).

The camera intrinsic matrix can be multiplied with any 3D point $\mathbf{X} = [x, y, z]^\top$ to determine its homogeneous coordinates. By utilising homogeneous coordinates the 'warping' process $\mathcal{W}$ that transfers this 3D point to 2D image coordinates is then

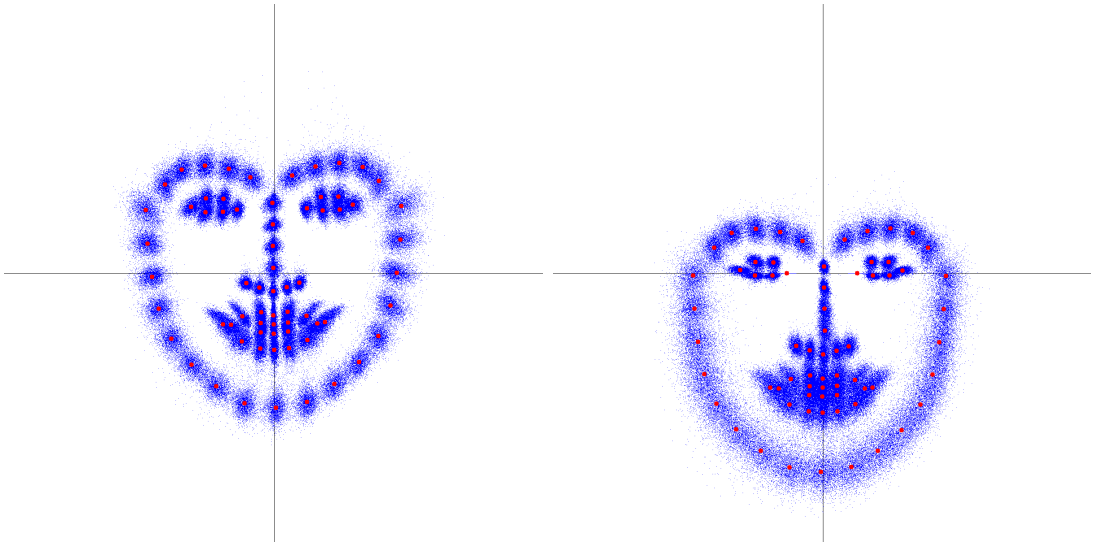
$$\mathcal{W}(\mathbf{X}) = E(\mathbf{K}\mathbf{X}) \tag{4.7}$$

4.2.3 Constructing the shape model

The shape model is based on the well known Point Distribution Model (PDM), where the 3D shape is defined as a vector $\mathbf{s}$

$$\mathbf{s} = \begin{bmatrix} x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n, z_1, z_2, \ldots, z_n \end{bmatrix}^\top \tag{4.8}$$

The shape $\mathbf{s}$ is made up on n individual vertex points in space located at suitable locations on the head such as the nose tip and eye corners. For the



(a) Generalised Procrustes alignment (b) Inner eye corner alignment

Fig. 4.4 Visualisation of the 3D training shapes (*xy-plane* shown) aligned via a Generalised Procrustes method and a new method aligned relative to the inner eye corners. Note how the statistical point distribution around the eyes becomes tightly packed with the inner eye corners only able to move symmetrically along the *x-axis*.

creation of a 3D parametric model it is necessary to capture the training data as actual 3D data measured in millimetres (mm) to give us the benefits of obtaining real-world pose estimation values. This is directly opposed to more traditional 2D approaches where tagged images are required to be scaled, translated and rotated via a Generalised Procrustes alignment and is frequently carried needlessly to the 3D case (Baltrušaitis et al., 2012; Martins et al., 2012; Saragih et al., 2011). The origin of the model then becomes the mean of all points as shown in Figure (4.4a) and implies that a deformation of the model can change the pose which is unsatisfactory. One of the novel contributions of this work is that the 3D models are instead aligned with a predefined 'origin' of the face located midway between the inner eye corners and rotated such that the inner eye corners lie on the *x-axis* and the underside of the nose lies on the *y-axis*, as shown in Figure (4.4b). This is important since it ensures that all pose estimations are related to a fixed point in the eye region and crucially, prevents the more deformable parts of the face such as the mouth from negatively affecting the pose estimation.

The 3D training data can be acquired directly, for example, by tagging ver-

tices in 3D modelling software on data captured from a range scanner which measures depth through lasers or other means. It can also be acquired indirectly though methods such as Non-Rigid Structure from Motion (NRSfM) (Torresani et al., 2008) whereby the 3D data is estimated from several tagged 2D images simultaneously captured of the object in question (Saragih et al., 2011). Due to the time needed to scan and tag 3D points directly there is a limited amount of datasets available for research purposes, although with the introduction of cheap 3D cameras such as *Microsoft Kinect* and *Intel Realsense* this is likely to change in the coming years.

The objective is to obtain a linearised parametric model of the head,

$$\mathbf{s} = \mathbf{s}_0 + \mathbf{d}\mathbf{\Phi} \tag{4.9}$$

where $\mathbf{s}_0$ is the mean shape and $\mathbf{\Phi}$ a set of ν orthogonal linear basis vectors (eigenvectors) describing the directions in which the shape can deform, parameterised by weighted values $\mathbf{d} = \{d_1, d_2, \ldots, d_\nu\}$. To obtain the mean-shape and eigenvectors a Principal Component Analysis (PCA) method is applied to the training shape vectors once the data has been rotated and translated (not scaled) via the eye corners and nose (Figure 4.5). Additionally the process obtains the equivalent eigenvalues $\mathbf{\Lambda}$ which give the statistical variance along the vectors. These values can act as limits for how far the eigenvectors are allowed to stretch and squash the model. Only 95% of the variability is retained due to the likelihood of small variances from misplaced tags.

The model is completed with a set of 6 pose parameters $\mathbf{p} = [r_x, r_y, r_z, t_x, t_y, t_z]^\top$ which describe the 6 degrees of freedom for the position of the head in the world derived in section (4.2.1). In particular, the rotation parameters allow for the subsequent calculation of the 3×3 matrix $\mathbf{R}$ (using equation 4.2).

With parameters $\mathbf{d}$ and $\mathbf{p}$, the k^{th} 3D point of the shape $\mathbf{X}_k = [x_k, y_k, z_k]^\top$

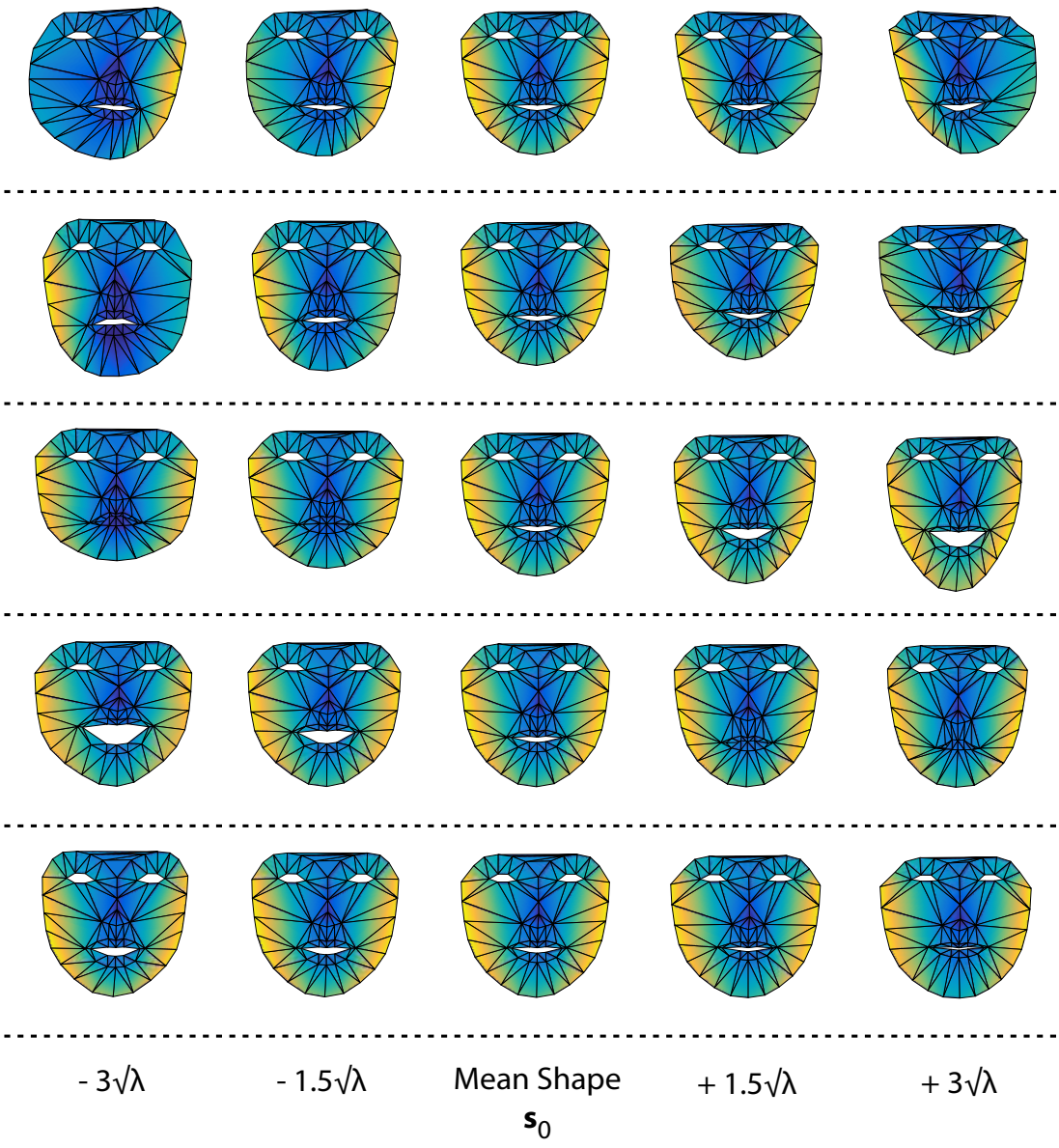


Fig. 4.5 The first 5 principal components of the head model are shown with each row representing an eigenvector. The output is constrained to within 3 standard deviations to ensure a realistic face shape is produced. All shapes have their origin equidistant between the inner eye corners which lie on the *x-axis*. Note that since the cheeks are not well-defined, the first principal component has to take into account a large range of possible cheek outlines.

can be determined in world space as

$$\mathbf{X}_k(\mathbf{d},\mathbf{p}) = \mathbf{R}\begin{bmatrix} \mathbf{s}_0^{x_k} + \sum_{i=1}^{\nu} d_i \mathbf{\Phi}_i^{x_k} \\ \mathbf{s}_0^{y_k} + \sum_{i=1}^{\nu} d_i \mathbf{\Phi}_i^{y_k} \\ \mathbf{s}_0^{z_k} + \sum_{i=1}^{\nu} d_i \mathbf{\Phi}_i^{z_k} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{4.10}$$

Using the warping function $\mathcal{W}$ (equation 4.7), all the points of the model can be located within the image using a whole model projection $\mathcal{P}(\mathbf{d},\mathbf{p})$

$$\mathcal{P}(\mathbf{d},\mathbf{p}) = \begin{bmatrix} \mathcal{W}(\mathbf{X}_1(\mathbf{d},\mathbf{p})) \\ \mathcal{W}(\mathbf{X}_2(\mathbf{d},\mathbf{p})) \\ \vdots \\ \mathcal{W}(\mathbf{X}_n(\mathbf{d},\mathbf{p})) \end{bmatrix} \tag{4.11}$$

4.3 CLM fitting objective

To fit our PDM to an image we need to look at finding the joint parameter values $\mathbf{d}$ and $\mathbf{p}$ that minimise the misalignment of each vertex in the shape model to the estimated image location $\mathbf{x}$ for that vertex. Concatenating the pose and deformation parameter vectors to form a new vector $\mathbf{q} = [\mathbf{p}|\mathbf{d}]$, we define the error function $\mathcal{E}$ as

$$\mathcal{E}(\mathbf{q}) = \mathcal{R}(\mathbf{q}) + \sum_{i=1}^{n} \mathcal{D}_i(\mathbf{x}_i;\mathcal{I}) \tag{4.12}$$

where $\mathcal{D}_i$ is a function representing the incoming *data* and evaluates how each landmark $\mathbf{x}_i$ is misaligned within the image $\mathcal{I}$. Note as in (Saragih et al., 2011) the extension of the error term with a *regularisation* term $\mathcal{R}$ in place to punish complex deformations of the shape model (i.e. shapes that deviate too far from the mean shape). The following sections will discuss the implementations of both the *regularisation* and *data* terms in more detail.

4.3.1 A Maximum A-Posteriori equivalent

An alternative way of viewing the problem is via a probabilistic interpretation (Saragih et al., 2011). Assuming conditional independence of all the landmarks let us observe that the probability (p) of the shape model being correctly aligned with $\mathbf{q}$ parameters within the image $\mathcal{I}$ is proportional to the product of each individual landmark probability being correctly aligned at $\mathbf{x}_i$ as follows

$$p(\mathbf{q}|\{l_i = 1\}_{i=1}^n, \mathcal{I}) \propto \prod_{i=1}^{n} p(l_i = 1 \,|\mathbf{x}_i, \mathcal{I}) \tag{4.13}$$

where $l_i \subset \{-1, 1\}$ is a discrete random variable describing whether the i^{th} landmark is correctly aligned or not.

Our goal is to obtain the parameters $\hat{\mathbf{q}}$ that maximise the likelihood our model is correctly aligned within the image. One way of obtaining such parameters can be via a Maximum-Likelihood (ML) estimation

$$\hat{\mathbf{q}}_{ML} = \underset{\mathbf{q}}{\operatorname{argmax}} \left\{ \prod_{i=1}^{n} p(l_i = 1 \,|\mathbf{x}_i, \mathcal{I}) \right\} \tag{4.14}$$

Unfortunately, a problem with ML estimation is that the variance of the parameter estimates can be high i.e. it 'overfits' the model to the incoming data and thus is very sensitive to noise of which there is often a substantial amount within computer vision problems.

To overcome the overfitting problem a regularisation bias can be added giving us a Maximum A-Posteriori (MAP) solution. This bias is provided in the form of our prior beliefs of how the $\mathbf{q}$ parameters vary, $p(\mathbf{q})$.

$$\hat{\mathbf{q}}_{MAP} = \underset{\mathbf{q}}{\operatorname{argmax}} \left\{ p(\mathbf{q}) \prod_{i=1}^{n} p(l_i = 1 \,|\mathbf{x}_i, \mathcal{I}) \right\} \tag{4.15}$$

In order to make this equation match the terms in the original fitting objective (equation 4.12) the log-likelihood can be taken. By taking the log-likelihood (which provides the equivalent parameters since it is a monotonically increasing function) the individual probabilities can instead be summed and it can be seen

that

$$\begin{aligned}\hat{\mathbf{q}}_{MAP} &= \underset{\mathbf{q}}{\operatorname{argmax}} \left\{ \ln \left\{ p(\mathbf{q}) \prod_{i=1}^{n} p(l_i = 1 \,|\mathbf{x}_i, \mathcal{I}) \right\} \right\} \\ &= \underset{\mathbf{q}}{\operatorname{argmax}} \left\{ \ln \{p(\mathbf{q})\} + \ln \left\{ \prod_{i=1}^{n} p(l_i = 1 \,|\mathbf{x}_i, \mathcal{I}) \right\} \right\} \\ &= \underset{\mathbf{q}}{\operatorname{argmax}} \left\{ \ln \{p(\mathbf{q})\} + \sum_{i=1}^{n} \ln \{p(l_i = 1 \,|\mathbf{x}_i, \mathcal{I})\} \right\} \end{aligned} \tag{4.16}$$

Taking its negative gives us a function to minimise

$$\hat{\mathbf{q}}_{MAP} = \underset{\mathbf{q}}{\operatorname{argmin}} \left\{ -\ln \{p(\mathbf{q})\} + \sum_{i=1}^{n} -\ln \{p(l_i = 1 \,|\mathbf{x}_i, \mathcal{I})\} \right\} \tag{4.17}$$

which is equivalent to our original formulation of the CLM where the *Regularisation* and *Data* terms are now

$$\mathcal{R}(\mathbf{q}) = -\ln \{p(\mathbf{q})\} \tag{4.18}$$

$$\mathcal{D}_i(\mathbf{x}_i; \mathcal{I}) = -\ln \{p(l_i = 1 \,|\mathbf{x}_i, \mathcal{I})\} \tag{4.19}$$

4.3.2 Evaluation of the *regularisation* term

The *regularisation* term is dependent on the probability of our prior beliefs about the shape model, and thus our problem becomes *'Given a collection of n ordered 3D points, how likely is it they represent a face shape?'* The rigid component of the **q**-parameters **p** (representing the pose) can be any translation and rotation so a non-informative prior can be used. As such the regulariser only depends on the deformation parameters **d**.

Fortunately, our knowledge of the shape is inherent in its construction using PCA. This process defined a linear parametric model with a mean shape and ν orthogonal eigenvectors $\mathbf{\Phi}$ representing direction of allowed movement. These eigenvectors have respective eigenvalues $\mathbf{\Lambda}$ representing statistical variance along those vectors, and thus we have a multivariate Gaussian distribution in ν dimen-

sions.

$$p(\mathbf{q}) \sim \mathcal{N}(\mathbf{d}; \mathbf{\Lambda}); \qquad \mathbf{\Lambda} = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_\nu\} \tag{4.20}$$

where λ_i denotes the eigenvalue for the i^{th} deformation vector and

$$\mathcal{N}(\mathbf{d}; \mathbf{\Lambda}) = \frac{1}{\sqrt{(2\pi)^\nu \, |\mathbf{\Lambda}|}} \, e^{-0.5 \, \left(\mathbf{d}^T \mathbf{\Lambda}^{-1} \mathbf{d}\right)} \tag{4.21}$$

The *regularisation* term can now be simplified as

$$\begin{aligned} \mathcal{R}(\mathbf{q}) &= -\ln\{p(\mathbf{q})\} \\ &\propto -\ln\left\{ \frac{1}{\sqrt{(2\pi)^\nu \, |\mathbf{\Lambda}|}} \, e^{-0.5 \, \left(\mathbf{d}^T \mathbf{\Lambda}^{-1} \mathbf{d}\right)} \right\} \\ &\propto -\ln\left\{ \frac{1}{\sqrt{(2\pi)^\nu \, |\mathbf{\Lambda}|}} \right\} - \ln\left\{ \, e^{-0.5 \, \left(\mathbf{d}^T \mathbf{\Lambda}^{-1} \mathbf{d}\right)} \right\} \\ &\propto \ln\left\{ \sqrt{(2\pi)^\nu \, |\mathbf{\Lambda}|} \right\} + 0.5 \, \left(\mathbf{d}^T \mathbf{\Lambda}^{-1} \mathbf{d}\right) \end{aligned} \tag{4.22}$$

Since $\ln\left\{ \sqrt{(2\pi)^\nu \, |\mathbf{\Lambda}|} \right\}$ is a constant term it has no bearing on the minimisation result and can be discarded. Hence we are left with

$$\mathcal{R}(\mathbf{q}) = r \, \|\mathbf{d}\|^2_{\mathbf{\Lambda}^{-1}} \tag{4.23}$$

where r is a regularisation weight of our choosing and $\|\mathbf{d}\|^2_{\mathbf{\Lambda}^{-1}}$ is shorthand for the squared Mahalanobis distance $\mathbf{d}^T \mathbf{\Lambda}^{-1} \mathbf{d}$. Note that when our current shape estimate is set to the mean shape, the deformable parameters are $\mathbf{0}$ and hence the regularisation term is 0. The further the shape deforms from the mean shape, the larger the regularisation penalty becomes. The value r allows us to bias the result in favour of either the prior knowledge or the incoming observed data and as suggested by Saragih et al. (2011) setting r to the mean of the eigenvalues

$$r = \frac{1}{\nu} \sum_{i=1}^{\nu} \lambda_i \tag{4.24}$$

empirically produces stable results on the trained face model.

For clarity and to help solve for the parameters it is helpful to keep the equation in terms of $\mathbf{q}$

$$\mathcal{R}(\mathbf{q}) = r \left\| \mathbf{q} \right\|^2_{\tilde{\mathbf{\Lambda}}^{-1}} \tag{4.25}$$

where a non-informative prior is kept on the first 6 parameters representing the pose

$$\tilde{\mathbf{\Lambda}}^{-1} = \text{diag}\{0, 0, 0, 0, 0, 0, \frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_\nu}\} \tag{4.26}$$

4.3.3 Evaluation of the *data* term

The *data* term $\mathcal{D}$ represents the incoming information from the camera and is specifically tasked in finding the most likely position of the head within the image frame (assuming there is one within the image). To accomplish this, it is important to know what we are looking for and to minimise false positives.

Since each landmark can be evaluated independently, the error term each landmark contributes to the overall error term in its simplest form can be obtained simply through calculating the squared Euclidean distance from the projected shape model points to the observed optimal locations $\mathbf{y}$ within the image.

$$\mathcal{D}(\mathbf{x}, \mathcal{I}) = \sum_{1}^{n} (\mathbf{y}_i - \mathcal{W}(\mathbf{X}_i, \mathbf{q}))^2 = \left\| \mathbf{y} - \mathcal{P}(\mathbf{q}) \right\|^2 \tag{4.27}$$

While it is trivial here to apply weights to individual points to reflect their importance to the overall shape fitting, in this work all of the points are weighted equally other than when the points are outside the image frame, in which case they do not contribute to the overall error.

Combining equations (4.25) and (4.27) gives us

$$\begin{aligned} \hat{\mathbf{q}}_{MAP} &= \underset{\mathbf{q}}{\text{argmin}} \{\mathcal{R}(\mathbf{q}) + \mathcal{D}(\mathbf{x}, \mathcal{I})\} \\ &= \underset{\mathbf{q}}{\text{argmin}} \left\{ r \left\| \mathbf{q} \right\|^2_{\tilde{\mathbf{\Lambda}}^{-1}} + \left\| \mathbf{y} - \mathcal{P}(\mathbf{q}) \right\|^2 \right\} \end{aligned} \tag{4.28}$$

Since this minimisation formulation is in the form of a non-linear least squares problem, the problem is solved incrementally with linear update approximations added to the current parameters: $\mathbf{q} \leftarrow \mathbf{q} + \boldsymbol{\Delta}\mathbf{q}$ (with a slight adjustment for the rotation parameters - see section 4.6.3). Taking a first order Taylor series expansion of the projected landmarks we obtain

$$\mathcal{P}(\mathbf{q} + \boldsymbol{\Delta}\mathbf{q}) \approx \mathcal{P}(\mathbf{q}) + \mathbf{J}\boldsymbol{\Delta}\mathbf{q} \tag{4.29}$$

where $\mathbf{J}$ is the Jacobian matrix containing the derivatives of the projected shape points $\mathcal{P}(\mathbf{q})$ with respect to the $\mathbf{q}$ parameters i.e.

$$\mathbf{J} = \frac{\partial \mathcal{P}(\mathbf{q})}{\partial \mathbf{q}} \tag{4.30}$$

Thus we are looking to minimise

$$\widehat{\boldsymbol{\Delta}\mathbf{q}} = \underset{\boldsymbol{\Delta}\mathbf{q}}{\operatorname{argmin}} \left\{ r \left\| \mathbf{q} + \boldsymbol{\Delta}\mathbf{q} \right\|^2_{\tilde{\boldsymbol{\Lambda}}^{-1}} + \left\| \mathbf{y} - (\mathcal{P}(\mathbf{q}) + \mathbf{J}\boldsymbol{\Delta}\mathbf{q}) \right\|^2 \right\} \tag{4.31}$$

Of course, the real difficulty is now trying to determine the optimal locations $\mathbf{y}$, which we shall see in the following sections.

4.4 Collecting texture information

When developing a model for tracking it is imperative that the texture information we are using is truly representative of the underlying shape model. Since generative textures often do not adapt well to differing environments and people, the problem is approached from a discriminative perspective. Small texture patch information is collected that describes points on the given 3D shape model and allows for efficient computation. The image location has to match as closely as possible with the observed location of the 3D shape point, which of course is very difficult for simple ellipsoidal or cylindrical head models since they are only estimates of the underlying shape. Even acquiring representative information for the more accurate PDM can be difficult, since annotating face markers is a sub-

jective process and is ill-defined. For example, objectively annotating the tip of the nose can be extremely difficult on a 2D image.

Another issue regardless of how complex the underlying shape model, is the restriction on how well a 2D texture can represent a 3D object from many different angles. If our model only uses a limited amount of well-defined points, even small changes in texture due to rotation can become the source of large errors or loss of tracking completely. With this in mind, texture data is collected from a variety of different angles representing the different viewpoints the users head may be realistically seen from by the camera.

Like many machine learning algorithms, creating a general pattern matching algorithm requires variety within the dataset, reflecting a diverse range of people with differing face characteristics such as eye and nose shape, eyebrow thickness and colour, skin tone and whether or not they wear glasses.

Generally, texture information can be collected for training purposes either through hand-annotation or through tagging from an automated computer algorithm. The naive approach is perhaps to hand-annotate a collection of 2D images. This is a laborious process and would take a great deal of time to collect. Computer algorithms on the other hand are very fast but can be prone to large errors. Fortunately there are now various image datasets available to academics with annotated data that has been tested thoroughly such as the MultiPIE dataset (Gross et al., 2010).

4.4.1 2.5D CLM texture collection

For a successful 3D shape tracker that is used for the purposes of eye gaze tracking, face points should be chosen that:

- tend to remain static, so as to limit the amount of deformation of the model
- describe the region around the eyes well
- have surrounding textures that are *well defined*, i.e. limit the reliance of landmarks that are defined by shadows or outlines.

The last point about outlines in the image is important to emphasise. Many models observed within the literature do not account for the fact that while a landmark such as a persons cheek on the outline of their face are relatively easy to detect, they do not represent a single 3D point on the face, but instead a whole region of possible points as the user orients their head about in space. This can lead to large variations from the possible deformations and can hinder the pose estimation process. The first eigenvector of the face shown in Figure (4.5) shows this clearly, with large deviations in the cheeks representing the single biggest deformation parameter of the model.

On the other hand, the larger the number of landmarks that are tracked, the less likely the model will be thrown off by erroneous landmark estimations. It is also important to consider the trade-off in accuracy and efficiency. Additionally, acquiring hand-annotated coordinates for a large number of images is a tiresome task and is certainly an intractable method for individual researchers to build a large and varied dataset.

4.4.2 Head shape

Due to the difficulty in obtaining a large number of hand-annotated images, research institutions have funded the creation of large datasets which have been shared with the research community. Unfortunately, tagged 3D points for faces are not available due to the difficulty in acquiring high accuracy data. In order to acquire a large variety of 3D data required it has been shown that Non-Rigid Structure from Motion (NRSfM) techniques (Torresani et al., 2008) applied to the Multi-PIE (Gross et al., 2010) dataset can provide a suitable alternative (Saragih et al., 2011).

The NRSfM MATLAB implementation by Torresani et al. (2008) is applied to the 68 annotated points within the images in order to obtain the 3D vertex information. Although there is a small amount of noise from the process due to the fact that it is difficult to hand-select identical points from multiple images, the algorithm does a good job in capturing the variability of the faces in the dataset, and produces believable face representations.

There is a clear benefit here from the change in shape alignment for the PCA

process. When keeping 95% of the shape variation from the training set, using Procrustes alignment the shapes produce a deformable model with 24 eigenvectors. The alternative alignment (described in section 4.2.3) produces a PDM with only 14 eigenvectors, massively simplifying the calculations required to optimise the shape.

4.5 Texture patch filters

The local patch filters work independently at first and have a simple goal - to search their local image space and provide a probability of how well the pixels in the vicinity resemble the tagged training sample, with closer resemblances achieving a higher response rate. A pixel here in fact represents the centre of a sliding window of some width and height, and therefore is ideally suited for convolution or cross-correlation which can be efficiently applied within the Fourier domain. The model itself does not care which discriminator is used and so several different types can easily be substituted in and compared for speed, accuracy and robustness.

In this work two different discriminators are trialled and compared. First a Local Neural Field (LNF) that uses positive image samples along with negative samples taken from areas close to the the correct area to find a non-linear mapping between the image data and the ideal response (Baltrusaitis et al., 2013). A number of 'neurons' are generated where each pixel is associated with a weight $\mathbf{w}$ that along with a bias term b attempt to discriminate between the positive and negative samples. Each neuron has to be convolved with the incoming image test sample

$$\Gamma_i(\mathbf{x}, \mathcal{I}) = \mathbf{w_i}^T \mathcal{V}(\hat{\mathbf{x}}, \mathcal{I}_i) + b_i \tag{4.32}$$

where $\mathcal{V}(\hat{\mathbf{x}}, \mathcal{I}_i)$ is a vectorised window image patch around the image point $\hat{\mathbf{x}}$ altered to zero mean and unit variance. The final response is then obtaining by summing each neuron's individual response. The window is a box centred at $\mathbf{x}$ that can search as much of the local area as is required. The LNF tested in this work is freely available as part of the OpenFace framework (Baltrušaitis et al., 2016). An example of the implementation can be seen in Figure (4.6) that has

7 neurons with each having a support region of only 11x11 pixels. For further details on the training and implementations details see (Baltrusaitis et al., 2013; Baltrušaitis et al., 2016).

Fig. 4.6 The LNF comprises of multiple filters for a single point of the PDM (here showing the outer eye corner). Each of the filters need to be convolved with the new input image before their responses are summed. This makes them more robust, although there is a cost in computation time.

The other discriminative texture filter implemented is based on a MOSSE (Minimum Output Sum of Squared Error) filter (Bolme et al., 2010), which act naturally in the Fourier domain. The MOSSE filters are capable of overcoming such issues as minor variances in rotation and large changes in lighting, even with a small amount of training data. As such they tend to show good robustness to the environment variations that are likely to be observed on mobile devices while still maintaining very fast computation times due to their efficient point-wise multiplication (denoted $\odot$) at run-time.

Fig. 4.7 MOSSE Filter Examples

The MOSSE filter detector output Γ_i for each new image patch $\mathcal{I}_i$ around the region of point i is given by:

$$\Gamma_i = \mathscr{F}^{-1}(\mathscr{F}(\mathcal{I}_i) \odot \mathcal{H}_i^*) \tag{4.33}$$

with the trained filter $\mathcal{H}_i^*$ calculated over n training patch samples

$$\mathcal{H}_i^* = \frac{\sum_{j=1}^{n} \mathcal{G}_j \odot \mathscr{F}(\mathcal{I}_j)^*}{\sum_{j=1}^{n} \mathscr{F}(\mathcal{I}_j) \odot \mathcal{F}(\mathcal{I}_j)^*} \tag{4.34}$$

$\mathscr{F}$ denotes a transfer to the Fourier domain. $\mathcal{G}$ is the desired output image

with peak response as a Gaussian with 2 pixels of standard deviation, and * denotes the complex conjugate.

4.5.1 Adapted MOSSE filter

The MOSSE filter in its original form best operates with image patch sizes that are a power of 2 as this is when transfer into the Fourier domain is most efficient (Cooley and Tukey, 1965). This places an additional restriction on the possible filter sizes. In addition, the filter and search region must be of equal size to perform the point-wise multiplication.

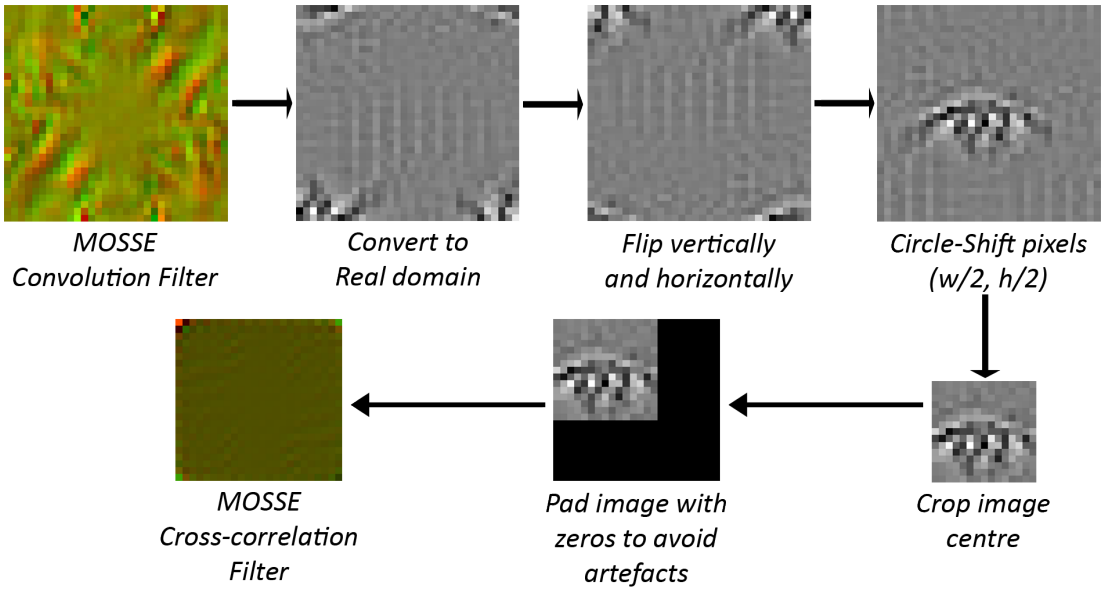


Fig. 4.8 Post-processing the MOSSE filters allows the filters to be cropped to different sizes.

To combat these issues, the MOSSE convolution filters are refactored into cross-correlation filters by flipping the final MOSSE filters vertically and horizontally in the spatial domain and shifting the filter such that the origin is no longer in the centre but at the top-left hand corner. The filter can then be cropped to any required size. Since calculations in the Fourier domain involve a circle-rotation of the input image, artefacts can be produced at the edges. To avoid this and to maintain the power of 2 requirement the filter can be zero-padded (this remains computationally efficient due to the Discrete Fourier Transform implementation used from the OpenCV library (Bradski, 2000), which allows for

setting the number of rows to use in the computation). The final filter $\hat{\mathcal{H}}$ can then be converted back to the Fourier domain and stored within a data file saving the need to compute it in real time. The cross-correlation is then given by

$$\Gamma_i = \mathscr{F}^{-1}(\mathscr{F}(\mathcal{I}_i) \odot \hat{\mathcal{H}}_i) \tag{4.35}$$

A diagram describing this post-processing can be seen in Figure (4.8).

4.5.2 Patch training

To train the patch filters, the Multi-PIE training set (Gross et al., 2010) is again used since it has a wide variety of illumination conditions as well as variability in people and facial expressions. The Multi-PIE dataset also has another big advantage in that it simultaneously captures images from various views. To accommodate different head orientations (where the texture patch can fluctuate significantly), 9 different sets of patches are trained which capture the likely observed head angles from the camera. These are

$$\text{yaw} = \{-90°, -75°, -45°, -20°,\ 0°,\ 20°,\ 45°,\ 75°,\ 90°\} \tag{4.36}$$

The patch data is collected at some standard set size so that the image patch we are looking for can be identified during run-time. The standard approach (Baltrušaitis et al., 2012; Martins et al., 2012; Saragih et al., 2011) is to adjust the incoming patch via a Procrustes alignment which aligns the model through scale, rotation and translation of the observed 2D projected points. As the head rotates and is observed from different angles this has the effect of actually changing the overall size of the head which can cause the model to 'jump' when switching between different trained patch sets. A novel feature of this work is ensuring that the transitions between patch sets is smooth as it does not depend on a Procrustes alignment of the 2D points but through a simple 3D projection technique.

The goal is to rotate the incoming image to make the face appear as 'upright' as possible. This is done by creating a 3D circle on the local *xz-plane* of the model, as shown in Figure (4.10) and trying to align the circle so that it becomes as flat

Fig. 4.9 Face alignment examples from the LFPW Dataset (Belhumeur et al., 2013). The images have been rotated and scaled using a projection technique which allows for smooth shifting between multiple viewpoints. Note in particular the eye alignment remains approximately horizontal and correctly placed from all angles.

as possible in the image. In all but the most trivial cases it will be impossible to rotate the image such that the circle becomes completely horizontal, so a heuristic is introduced that produces suitable and consistent results.

Let the 3D vectors of the circle be defined in $\mathbb{R}^3$ by its centre $\mathbf{O}$ and axes $\mathbf{U}$ and $\mathbf{V}$ where

$$\begin{aligned} \mathbf{O} &= [0,0,0]^\top \\ \mathbf{U} &= [0,0,-1]^\top \\ \mathbf{V} &= [1,0,0]^\top \end{aligned} \tag{4.37}$$

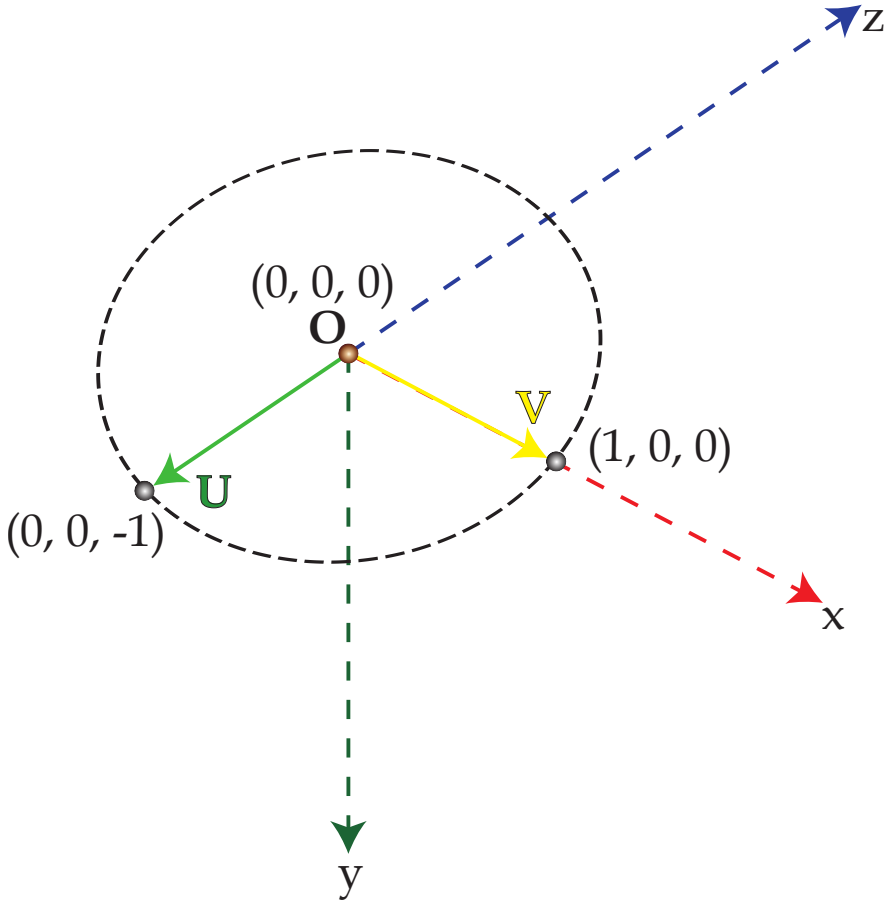


Fig. 4.10 A circle defined in 3D space with its centre at the origin of the head model. When projected into an image at the current head pose it provides a heuristic to rotate the image to a standard orientation.

The vectors can be determined in the image space $\mathbb{R}^2$ through a projection using the camera intrinsic matrix and the current pose parameters $\mathbf{p}$. These new vectors are defined

$$\begin{aligned} \mathbf{u} &= \mathcal{W}(\mathbf{U}, \mathbf{p}) - \mathcal{W}(\mathbf{O}, \mathbf{p}) \\ \mathbf{v} &= \mathcal{W}(\mathbf{V}, \mathbf{p}) - \mathcal{W}(\mathbf{O}, \mathbf{p}) \end{aligned} \tag{4.38}$$

For front facing images we wish to rotate the patch such that the eyes are horizontal. However, if the head is closer to a side-on perspective this is not suitable since the perspective camera typically places one eye above the other. In these instances it is better to rotate 'from the chin' upwards until approximately upright. These vectors therefore act as two competing forces trying to rotate

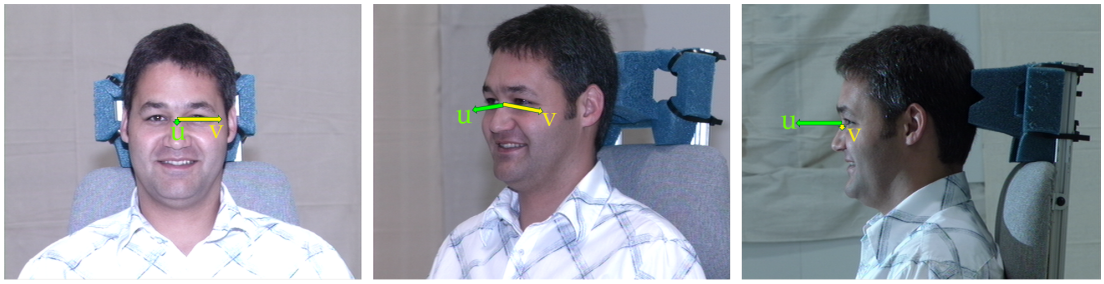


Fig. 4.11 Projected examples of the **u** and **v** vectors. Many cases will likely have two reasonable options to rotate the image about and so a weighted approach is taken based on the magnitude of the respective vector's horizontal component. When the user is facing toward the camera $u_H \rightarrow 0$. A side-on view makes $v_H \rightarrow 0$.

the image. To overcome this fact a weight can be applied to each to allow for a gradual shift between the two influences as the user rotates their head. These weights w_u and w_v are calculated depending on their current magnitude of their horizontal component within the image as

$$\begin{aligned} w_u &= \sin^2\left(0.5\pi \frac{|u_H|}{|u_H| + |v_H|}\right) \\ w_v &= 1 - w_u \end{aligned} \tag{4.39}$$

where u_H and v_H correspond to the horizontal dimension of the vectors. Three examples are shown with the projected u and v vectors in Figure (4.11). Note that when the user is facing toward the camera $u_H \rightarrow 0$ and $w_u \rightarrow 0$. A side-on view makes $v_H \rightarrow 0$ and therefore $w_u \rightarrow 1$.

The final image rotation α is

$$\alpha = w_u \arctan\left(\frac{u_y}{u_x}\right) + w_v \arctan\left(\frac{v_y}{v_x}\right) \tag{4.40}$$

The size of the face within the image is another factor that needs to be considered. Let o represent the real width of a 3D object and r represent its width in pixels within an image. The relationship between the two is defined as

$$r = \frac{of}{d} \tag{4.41}$$

where f is the camera focal length and d is the distance of the object from the camera. This relationship can be exploited to scale an incoming image to a standard reference size r_0 by evaluating its situated distance d_0 away from the camera. The scaling term s each frame is then evaluated as a ratio of the constant reference d_0 distance and the current distance of the head from the camera d

$$s = \frac{d_0}{d} \tag{4.42}$$

Now that the head is upright having undergone a 'roll' rotation we need to determine suitable 'yaw' (θ) and 'pitch' (ϕ) calculations of the head *as seen from the camera's perspective*. Determining the rotation pair $\boldsymbol{\Upsilon} = (\theta, \phi)$ will allow us to train patches from different angles, and then choose the most appropriate set of patches to use at runtime. There are 2 factors influencing these values - first the rotation of the head itself $(\theta_{rot}, \phi_{rot})$, and secondly, due to the perspective camera, the translation of the head away from the principal point of the camera $(\theta_{trans}, \phi_{trans})$. That is to say, as the head moves toward the outer regions of the image, the perceived angle of the head changes. The rotation pair as viewed from the camera is then defined as

$$\boldsymbol{\Upsilon} = (\theta_{rot} + \theta_{trans}, \quad \phi_{rot} + \phi_{trans}) \tag{4.43}$$

With the current pose of the head known by its rotation $\mathbf{R}$ and its translation $\mathbf{t}$ the pitch and yaw values can be obtained in the following way. Let $\mathbf{R}_\alpha$ be the inverse 3x3 rotation matrix of the angle α around the *z-axis* and again, let $U = [0, 0, -1]^\top$. The head yaw and pitch due to rotation are defined as

$$\begin{aligned} \mathbf{a} = [a_1, a_2, a_3] &= \mathbf{R}_\alpha \mathbf{R} \mathbf{U} \\ \theta_{rot} &= \arccos\left(\frac{-a_3}{\sqrt{{a_1}^2 + {a_3}^2}}\right) \\ \phi_{rot} &= \arccos\left(\frac{-a_3}{\sqrt{{a_2}^2 + {a_3}^2}}\right) \end{aligned} \tag{4.44}$$

For the yaw and pitch changes due to translation, we have

$$\begin{aligned} \mathbf{b} = [b_1, b_2, b_3] &= \mathbf{R}_\alpha \mathbf{t} \\ \theta_{trans} &= \arctan\left(\frac{b_1}{b_3}\right) \\ \phi_{trans} &= \arctan\left(\frac{b_2}{b_3}\right) \end{aligned} \tag{4.45}$$

Since the texture patches have been trained from multiple different camera perspectives (in this work 9), a distance metric is required to choose the closest training viewpoint to the incoming camera frame. For each training camera viewpoint $c \in \{1, ..., 9\}$ the yaw and pitch values are evaluated as in equation (4.43) and saved as $\boldsymbol{\tau}_c$ during patch training. The squared 'distance' value between the current orientation angles $\boldsymbol{\Upsilon}$ and each camera orientation $\boldsymbol{\tau}_c$ is evaluated as

$$\hat{c} = \underset{c}{\operatorname{argmin}} \left\{ \|\boldsymbol{\Upsilon} - \boldsymbol{\tau}_c\|^2 \right\} \tag{4.46}$$

with the smallest value belonging to the camera $\hat{c}$ with the closest match.

4.5.3 Patch size and scale

To train the filters, small patches were taken from the annotated training images and were affine warped to slightly varying scales, rotations, and translations to create a robust filter. For efficiency purposes the patch filters were kept small with the underlying image scaled to cover a region of the face large enough for discrimination. Additionally, the patches were trained at multiple different scales and patch sizes such that they could be used in a pyramid refinement sense starting with a wider search region with lower accuracy. Setting the default trained head width to 150 pixels, the trained regions and scales can be seen in Table 4.1.

To obtain the response in probabilistic terms to match our model, a logistic regressor is trained on each of the filter responses on the dataset. The regressor is calculated by taking the true positive match from the naive response of the filter $\Gamma_i(\mathbf{x}, \boldsymbol{\mathcal{I}})$ as well as many randomly chosen negative samples from misaligned

Size	Scale
11 x 11	1.0
9 x 9	0.8
7 x 7	0.6

Table 4.1 Trained patch sizes and scales, where a scale of 1.0 gives a head width of 150 pixels.

regions of the response maps. The regressor in turn estimates a gain α and bias β term to manipulate the response map results. At runtime the standard logistic regression curve then provides the likelihood of a correct match at a landmark $\mathbf{x}$

$$p(l_i = 1|\mathbf{x}, \mathcal{I}) = \frac{1}{1 + e^{-(\alpha_i \Gamma_i(\mathbf{x}, \boldsymbol{\mathcal{I}}) + \beta_i)}} \tag{4.47}$$

This has the effect of turning each likelihood into a probability mass function through a sigmoid curve which peaks at 1 for a positive match and 0 for a poor match.

4.6 Optimising the response

Since the response maps are determined from limited data, are small and have to combat large degrees of variation there is often noise and ambiguity with multiple peaks giving high response. While many previous CLMs optimised the response maps with simple parametric forms, Saragih et al. (2011) described how these were error-prone due to the multi-modal nature of the response map. Instead a non-parametric form was shown to give more accurate results in Regularised Landmark Mean-Shift (RLMS). The mean-shift algorithm relies on a Kernel Density Estimator (KDE) to gradually smooth the response map and 'push' the mean-shift vector toward the greatest concentration of high response points.

Allowing each data point in the response map $\mathbf{\Gamma}$ to exhibit some Gaussian

noise (with variance ρ), the data term becomes

$$\mathcal{D}_i = p(l_i = 1|\mathbf{x}_i, \mathcal{I}) = \sum_{\mathbf{y}_i \in \mathbf{\Gamma}_i} p(l_i = 1|\mathbf{y}_i, \mathcal{I})\, \mathcal{N}(\mathbf{x}_i, \mathbf{y}_i, \rho\mathbf{I}) \tag{4.48}$$

Normalising these posterior likelihoods gives us weighted values $w_{\mathbf{y}_i}$ where

$$w_{\mathbf{y}_i} = \frac{p(l_i = 1|\mathbf{y}_i, \mathcal{I})\, \mathcal{N}(\mathbf{x}_i, \mathbf{y}_i, \rho\mathbf{I})}{\sum_{\mathbf{z}_i \in \mathbf{\Gamma}_i} p(l_i = 1|\mathbf{z}_i, \mathcal{I})\, \mathcal{N}(\mathbf{x}_i, \mathbf{z}_i, \rho\mathbf{I})}; \qquad \sum_{\mathbf{y}_i \in \mathbf{\Gamma}_i} w_{\mathbf{y}_i} = 1 \tag{4.49}$$

The mean-shift vector $\mathbf{v}_i$ is then the normalised weighted pull $w_{\mathbf{y}_i}$ from each location in the response map $\mathbf{y}_i$ on the current estimate $\mathbf{x}_i$

$$\mathbf{v}_i = \left(\sum_{\mathbf{y}_i \in \mathbf{\Gamma}_i} w_{\mathbf{y}_i} \mathbf{y}_i \right) - \mathbf{x}_i \tag{4.50}$$

Saragih et al. (2011) showed that the mean-shift algorithm is equivalent to employing an Expectation Maximisation (EM) strategy which is a bound optimisation that converges to an optimal solution through iteration.

4.6.1 Solving for the new parameters

The optimal movement projections for all the points within the image frame can be stored within a matrix $\mathbf{v}$

$$\mathbf{v} = \mathbf{y} - \mathcal{P}(\mathbf{q}) \tag{4.51}$$

and therefore from equation (4.31) we are looking for the change in parameters $\widehat{\mathbf{\Delta q}}$ that minimise the following

$$\widehat{\mathbf{\Delta q}} = \underset{\mathbf{\Delta q}}{\operatorname{argmin}} \left\{ r \left\| \mathbf{q} + \mathbf{\Delta q} \right\|^2_{\tilde{\mathbf{\Lambda}}^{-1}} + \left\| \mathbf{v} - \mathbf{J}\mathbf{\Delta q} \right\|^2 \right\} \tag{4.52}$$

Since our problem is a non-linear least squares problem involving the sum of

square residuals the Gauss-Newton algorithm is used. Expanding the squared terms gives our error term as

$$\begin{aligned}\mathcal{E}(\mathbf{q}) &= r(\mathbf{q}+\boldsymbol{\Delta}\mathbf{q})^T\tilde{\boldsymbol{\Lambda}}^{-1}(\mathbf{q}+\boldsymbol{\Delta}\mathbf{q}) \\ &\quad + \mathbf{v}^T\mathbf{v} - 2\boldsymbol{\Delta}\mathbf{q}^T\mathbf{J}^T\mathbf{v} + \boldsymbol{\Delta}\mathbf{q}^T\mathbf{J}^T\mathbf{J}\boldsymbol{\Delta}\mathbf{q}\end{aligned} \tag{4.53}$$

which simplifies considerably when differentiating with respect to $\boldsymbol{\Delta}\mathbf{q}$

$$\frac{\partial\mathcal{E}(\mathbf{q})}{\partial\boldsymbol{\Delta}\mathbf{q}} = 2r\tilde{\boldsymbol{\Lambda}}^{-1}(\mathbf{q}+\boldsymbol{\Delta}\mathbf{q}) - 2\mathbf{J}^T\mathbf{v} + 2\mathbf{J}^T\mathbf{J}\boldsymbol{\Delta}\mathbf{q} \tag{4.54}$$

Setting this equal to 0 and solving for $\boldsymbol{\Delta}\mathbf{q}$

$$\begin{aligned}2r\tilde{\boldsymbol{\Lambda}}^{-1}(\mathbf{q}+\boldsymbol{\Delta}\mathbf{q}) - 2\mathbf{J}^T\mathbf{v} + 2\mathbf{J}^T\mathbf{J}\boldsymbol{\Delta}\mathbf{q} &= 0 \\ r\tilde{\boldsymbol{\Lambda}}^{-1}\mathbf{q} + r\tilde{\boldsymbol{\Lambda}}^{-1}\boldsymbol{\Delta}\mathbf{q} - \mathbf{J}^T\mathbf{v} + \mathbf{J}^T\mathbf{J}\boldsymbol{\Delta}\mathbf{q} &= 0 \\ \left(r\tilde{\boldsymbol{\Lambda}}^{-1} + \mathbf{J}^T\mathbf{J}\right)\boldsymbol{\Delta}\mathbf{q} + r\tilde{\boldsymbol{\Lambda}}^{-1}\mathbf{q} - \mathbf{J}^T\mathbf{v} &= 0 \\ \left(r\tilde{\boldsymbol{\Lambda}}^{-1} + \mathbf{J}^T\mathbf{J}\right)\boldsymbol{\Delta}\mathbf{q} &= \mathbf{J}^T\mathbf{v} - r\tilde{\boldsymbol{\Lambda}}^{-1}\mathbf{q}\end{aligned} \tag{4.55}$$

gives our parameter update as

$$\boldsymbol{\Delta}\mathbf{q} = \left(r\tilde{\boldsymbol{\Lambda}}^{-1} + \mathbf{J}^T\mathbf{J}\right)^{-1}\left(\mathbf{J}^T\mathbf{v} - r\tilde{\boldsymbol{\Lambda}}^{-1}\mathbf{q}\right) \tag{4.56}$$

4.6.2 Calculating the Jacobian

The Jacobian matrix ($\mathbf{J}$) from equation (4.30) is the derivative of the projected points in the shape model with respect to the $\mathbf{q}$ parameters.

$$\mathbf{J} = \left[\frac{\partial\mathcal{P}(\mathbf{q})}{\partial p_1} \quad \frac{\partial\mathcal{P}(\mathbf{q})}{\partial p_2} \quad \cdots \quad \frac{\partial\mathcal{P}(\mathbf{q})}{\partial p_6} \quad \frac{\partial\mathcal{P}(\mathbf{q})}{\partial d_1} \quad \frac{\partial\mathcal{P}(\mathbf{q})}{\partial d_2} \quad \cdots \quad \frac{\partial\mathcal{P}(\mathbf{q})}{\partial d_\nu}\right] \tag{4.57}$$

where

$$\frac{\partial \mathcal{P}(\mathbf{q})}{\partial p_i} = \begin{bmatrix} \frac{\partial \mathcal{W}(\mathbf{X}_1,\mathbf{q})}{\partial p_i} \\ \frac{\partial \mathcal{W}(\mathbf{X}_2,\mathbf{q})}{\partial p_i} \\ \vdots \\ \frac{\partial \mathcal{W}(\mathbf{X}_n,\mathbf{q})}{\partial p_i} \end{bmatrix}, \qquad \frac{\partial \mathcal{P}(\mathbf{q})}{\partial d_i} = \begin{bmatrix} \frac{\partial \mathcal{W}(\mathbf{X}_1,\mathbf{q})}{\partial d_i} \\ \frac{\partial \mathcal{W}(\mathbf{X}_2,\mathbf{q})}{\partial d_i} \\ \vdots \\ \frac{\partial \mathcal{W}(\mathbf{X}_n,\mathbf{q})}{\partial d_i} \end{bmatrix} \tag{4.58}$$

Intuitively, it describes how a small change in each of the parameters $\mathbf{q}_i$ is reflected in a change of the image coordinates. Notice that because the projection is non-linear, the derivative has the additional complexity that for each pose in 3D space, this 2D Jacobian is only locally applicable. This means the Jacobian matrix has to be calculated for each frame.

Combining equations (4.7) and (4.9) we know that the warp projection of the k^{th} 3D point of the shape $\mathbf{X}_k = [x_k, y_k, z_k]^\top$ to the 2D image plane $\mathbf{x}_k = [\hat{x}_k, \hat{y}_k]^\top$ with parameters $\mathbf{q}$ is defined as

$$\begin{aligned} \mathcal{W}(\mathbf{X}_k, \mathbf{q}) &= \begin{bmatrix} \hat{x}_k \\ \hat{y}_k \end{bmatrix} = E\left(\begin{bmatrix} w_k \hat{x}_k \\ w_k \hat{y}_k \\ w_k \end{bmatrix} \right) = E\left(\mathbf{K} \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} \right) \\ &= E\left(\begin{bmatrix} f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left(\mathbf{R} \begin{bmatrix} \mathbf{s}_0^{x_k} + \sum_{i=1}^{\nu} d_i \mathbf{\Phi}_i^{x_k} \\ \mathbf{s}_0^{y_k} + \sum_{i=1}^{\nu} d_i \mathbf{\Phi}_i^{y_k} \\ \mathbf{s}_0^{z_k} + \sum_{i=1}^{\nu} d_i \mathbf{\Phi}_i^{z_k} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right) \right) \end{aligned} \tag{4.59}$$

where the E operator converts the 3D homogeneous coordinates to 2D Euclidean coordinates (equation 4.5).

Ignoring the skew parameter α and any radial or tangential distortion, which

have a negligible effect on the derivative, the warp is approximately

$$\mathcal{W}(\mathbf{X}_k, \mathbf{q}) \approx \frac{1}{z_k} \begin{bmatrix} f_x x_k + c_x z_k \\ f_y y_k + c_y z_k \end{bmatrix} = \begin{bmatrix} \frac{f_x x_k}{z_k} + c_x \\ \frac{f_y y_k}{z_k} + c_y \end{bmatrix} \tag{4.60}$$

The derivatives for the warp can be calculated through the quotient rule

$$\begin{aligned} \frac{\partial \mathcal{W}(\mathbf{X}_k, \mathbf{q})}{\partial q_i} &= \begin{bmatrix} f_x \frac{\partial}{\partial q_i}\left(\frac{x_k}{z_k}\right) \\ f_y \frac{\partial}{\partial q_i}\left(\frac{y_k}{z_k}\right) \end{bmatrix} \\ &= \frac{1}{z_k^2} \begin{bmatrix} f_x \left(\frac{\partial(x_k)}{\partial q_i} z_k - x_k \frac{\partial(z_k)}{\partial q_i}\right) \\ f_y \left(\frac{\partial(y_k)}{\partial q_i} z_k - y_k \frac{\partial(z_k)}{\partial q_i}\right) \end{bmatrix} \end{aligned} \tag{4.61}$$

and hence with respect to each parameter in $\mathbf{q}$ we simply need to find the derivative of the 3D point and substitute it into equation (4.61).

The derivative of the point $\mathbf{X}_k$ defined in equation (4.10) with respect to the i^{th} deformable parameter $\mathbf{d}_i$ is simply the relevant eigenvectors rotated to match the model rotation.

$$\frac{\partial \mathbf{X}_k}{\partial d_i} = \begin{bmatrix} \frac{\partial(x_k)}{\partial d_i} \\ \frac{\partial(y_k)}{\partial d_i} \\ \frac{\partial(z_k)}{\partial d_i} \end{bmatrix} = \mathbf{R} \begin{bmatrix} \mathbf{\Phi}_i^{x_k} \\ \mathbf{\Phi}_i^{y_k} \\ \mathbf{\Phi}_i^{z_k} \end{bmatrix} \tag{4.62}$$

The derivative of the 3D point with respect to the pose parameters $\mathbf{p}$ is more complicated. Since we require a linear representation of a rotation around the current point, a matrix $\tilde{\mathbf{R}}$ representing an infinitesimally small rotation is introduced to the 3D model.

Since the rotation is so small (i.e. $\tilde{\theta} \to 0$) we can simplify the calculations by making the transformation linear using the fact that $\cos\tilde{\theta} \approx 1$ and $\sin\tilde{\theta} \approx \theta$.

The Rodrigues axis-angle rotation derived in equation (4.2) then becomes

$$\tilde{\mathbf{R}} \approx \mathbf{I} + \tilde{\theta} \begin{bmatrix} 0 & -\tilde{\omega}_z & \tilde{\omega}_y \\ \tilde{\omega}_z & 0 & -\tilde{\omega}_x \\ -\tilde{\omega}_y & \tilde{\omega}_x & 0 \end{bmatrix} = \begin{bmatrix} 1 & -\tilde{r}_z & \tilde{r}_y \\ \tilde{r}_z & 1 & -\tilde{r}_x \\ -\tilde{r}_y & \tilde{r}_x & 1 \end{bmatrix} \tag{4.63}$$

A 3D point on the model is defined as

$$\tilde{\mathbf{X}}_k(\mathbf{q}) = \mathbf{R}\tilde{\mathbf{R}} \begin{bmatrix} x^k_{\mathbf{d}} \\ y^k_{\mathbf{d}} \\ z^k_{\mathbf{d}} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \mathbf{R} \begin{bmatrix} x^k_{\mathbf{d}} - y^k_{\mathbf{d}}\tilde{r}_z + z^k_{\mathbf{d}}\tilde{r}_y \\ x^k_{\mathbf{d}}\tilde{r}_z + y^k_{\mathbf{d}} - z^k_{\mathbf{d}}\tilde{r}_x \\ -x^k_{\mathbf{d}}\tilde{r}_y + y^k_{\mathbf{d}}\tilde{r}_x + z^k_{\mathbf{d}} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{4.64}$$

where for clarity the k^{th} point before pose adjustments $(\mathbf{s}^k_0 + \mathbf{d}\mathbf{\Phi}^k_i) = [x^k_{\mathbf{d}}, y^k_{\mathbf{d}}, z^k_{\mathbf{d}}]^\top$.

Finally, the translation calculations are straightforward enough since they are real-numbered variables giving a derivative of 1 along each respective axis. The final derivatives can be summarised in the following way

$$\frac{\partial \tilde{\mathbf{X}}_k}{\partial \tilde{\mathbf{r}}} = \begin{bmatrix} \frac{\partial(x_k)}{\partial \tilde{r}_x} & \frac{\partial(x_k)}{\partial \tilde{r}_y} & \frac{\partial(x_k)}{\partial \tilde{r}_z} \\ \frac{\partial(y_k)}{\partial \tilde{r}_x} & \frac{\partial(y_k)}{\partial \tilde{r}_y} & \frac{\partial(y_k)}{\partial \tilde{r}_z} \\ \frac{\partial(z_k)}{\partial \tilde{r}_x} & \frac{\partial(z_k)}{\partial \tilde{r}_y} & \frac{\partial(z_k)}{\partial \tilde{r}_z} \end{bmatrix} = \mathbf{R} \begin{bmatrix} 0 & z^k_{\mathbf{d}} & -y^k_{\mathbf{d}} \\ -z^k_{\mathbf{d}} & 0 & x^k_{\mathbf{d}} \\ y^k_{\mathbf{d}} & -x^k_{\mathbf{d}} & 0 \end{bmatrix} \tag{4.65}$$

$$\frac{\partial \tilde{\mathbf{X}}_k}{\partial \mathbf{t}} = \begin{bmatrix} \frac{\partial(x_k)}{\partial t_x} & \frac{\partial(x_k)}{\partial t_y} & \frac{\partial(x_k)}{\partial t_z} \\ \frac{\partial(y_k)}{\partial t_x} & \frac{\partial(y_k)}{\partial t_y} & \frac{\partial(y_k)}{\partial t_z} \\ \frac{\partial(z_k)}{\partial t_x} & \frac{\partial(z_k)}{\partial t_y} & \frac{\partial(z_k)}{\partial t_z} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.66}$$

4.6.3 The rotation update

It is important to note here that while for the majority of parameters $\mathbf{q} \leftarrow \mathbf{q} + \mathbf{\Delta q}$ this is not actually correct for the rotation update because we are using a linear approximation of the rotation which is only valid for small rotations where $sin\theta \approx \theta$.

While the change in deformation parameters and translation parameters can simply be added together, successive rotations need to be multiplied to obtain the new rotation. To do this, the axis-angle rotation parameters are converted back to a rotation matrix $\mathbf{R}_\Delta$, and the new rotation $\mathbf{R'} = \mathbf{R}\mathbf{R}_\Delta$.

Unfortunately, this rotation matrix is no longer guaranteed to be orthogonal due to the linear approximation of the rotation eigenvectors. Similar to the work by Baltru
vsaitis (2014), a Singular Value Decomposition (SVD) is used that factorises the matrix into 2 orthogonal matrices $\mathbf{U}$ and $\mathbf{V^T}$ and a diagonal matrix $\mathbf{S}$.

$$\mathbf{U}\mathbf{S}\mathbf{V}^T = \text{SVD}(\mathbf{R}_\Delta) \tag{4.67}$$

The diagonal matrix can be discarded, giving an orthogonal rotation matrix

$$\hat{\mathbf{R}}_\Delta = \mathbf{U} \cdot \det(\mathbf{U}\mathbf{V}^T) \cdot \mathbf{V}^T \tag{4.68}$$

where $\det(\mathbf{U}\mathbf{V}^T) = \pm 1$, ensuring against the case of a reflection in the parameters. The final rotation matrix is then

$$\mathbf{R'} = \mathbf{R}\hat{\mathbf{R}}_\Delta \tag{4.69}$$

4.6.4 Face Alignment

On the initial frame or whenever we have lost tracking, the model is (re)initialised with a simple Haar classifier Viola and Jones (2001) which is a quick and reliable method of approximating the initial start region for face detection and is widely used in the literature.

Since during each frame only the local regions are searched there is a risk

that during large movements between frames (either through a low number of frames per second or high velocity of the user) the head may move out of the local search regions making tracking difficult. In an attempt to combat this, a global temporal movement tracker is built, providing a larger search region to determine approximate movement in x and y from the previous frame.

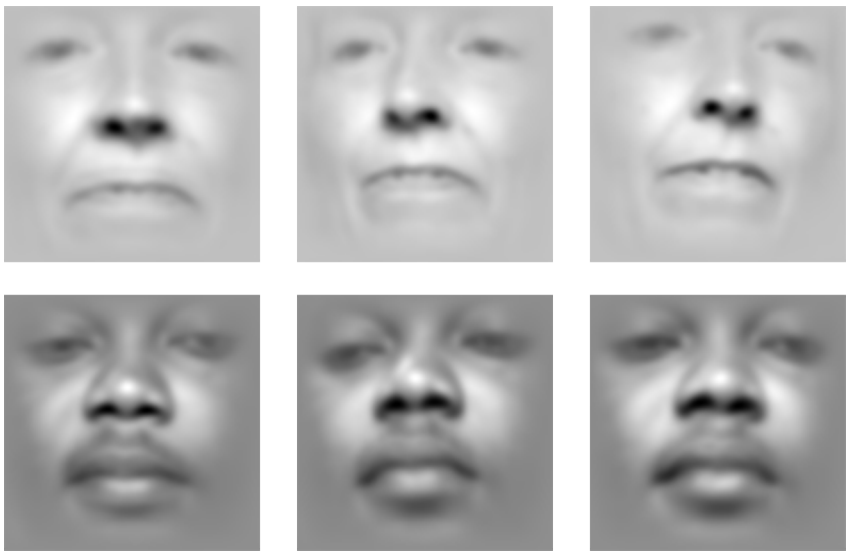


Fig. 4.12 The global tracker gradually adjusts to increase its robustness, particularly during rotations of the head.

Correlation filters are again highly suited to this kind of problem and can be implemented in linear (Bolme et al., 2010) and more recently non-linear (Henriques et al., 2015) forms. To limit computational cost a straightforward MOSSE filter is used but this time the model is not pre-trained but rather built on the fly. The region around the whole face is cropped and scaled to the correct size as discussed in section (4.5.2) giving a 128×128 patch model of the face. On the first frame during initialisation, 10 samples are taken by affine transforming the patch. Then as in Bolme et al. (2010) a learning rate is introduced (here denoted λ) that puts additional weight on more recent image data to let the filter template accommodate reasonable changes to the model that occur naturally under different environment conditions and pose changes. The original MOSSE equation (4.34) takes the form of a fraction, where both the numerator $\mathcal{A}$ and denominator $\mathcal{B}$ are calculated via a sum of all training samples. For the i^{th} frame

the constituent parts of the MOSSE filter can be stored separately as

$$\mathcal{H}_i^* = \frac{\mathcal{A}_i}{\mathcal{B}_i} \tag{4.70}$$

This allows the filter to be updated for each subsequent frame $(i + 1)$ as follows

$$\mathcal{A}_{i+1} = \lambda\, \mathcal{G}_{i+1} \odot \mathscr{F}(\mathcal{I}_{i+1})^* + (1 - \lambda)\mathcal{A}_i \tag{4.71}$$

$$\mathcal{B}_{i+1} = \lambda\, \mathscr{F}(\mathcal{I}_{i+1}) \odot \mathscr{F}(\mathcal{I}_{i+1})^* + (1 - \lambda)\mathcal{B}_i \tag{4.72}$$

A learning rate of $\lambda = 0.0125$ allows the filter to adapt quickly while remaining robust. The large size of the patch allows for a lot of detail within the filter, and since it is dynamic the peak response on correct tracking tends to be strong and dense. Incorrect tracking produces a weak and noisy response map. This allows for a simple measure to be used to detect tracking loss called the Peak to Sidelobe Ratio (PSR) (Bolme et al., 2010), which compares the maximum response value Γ_{MAX} with the mean μ_{SL} and standard deviation σ_{SL} of the response map when excluding a small window surrounding the peak.

$$PSR = \frac{\Gamma_{MAX} - \mu_{SL}}{\sigma_{SL}} \tag{4.73}$$

An example of the response map under successful and failed tracking can be seen in Figure (4.13). The PSR can vary widely depending on the complexities of the incoming image and the learning rate λ so the value chosen is specific to the implementation and can be adjusted depending on how important it is to avoid false positive scenarios. Figure (4.14) demonstrates the PSR under differing amounts of noise away from the true peak at the centre of the response map, with a strong second peak or large amount of noise decreasing the PSR significantly. In this work it was empirically observed that a value of below 15 suggested that the tracking was struggling when a 32×32 window was removed from the response map around the maximum point. To allow for short term obfuscation of the face the tracking is only deemed lost with consecutive failures over 10 frames. This triggers a new search from the Haar classifier and a reset of the model parameters.

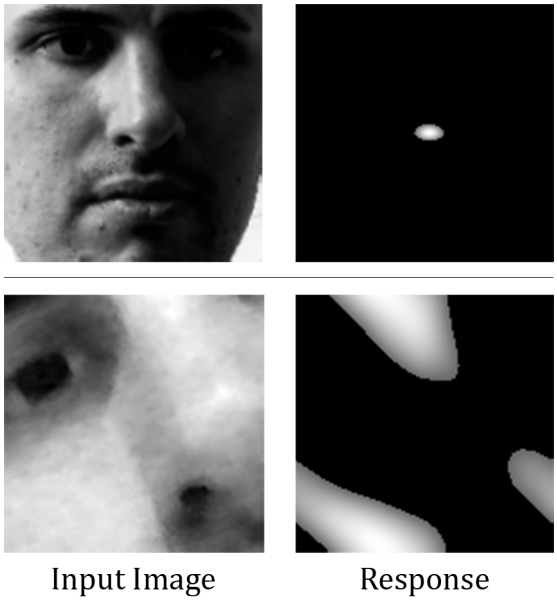


Fig. 4.13 Input images and their responses. The top row shows a strong PSR rate with strong response values isolated around the peak. The bottom row shows a failed tracking example - the PSR drops below 15 due to a noisy response map that does not have a strong isolated peak.

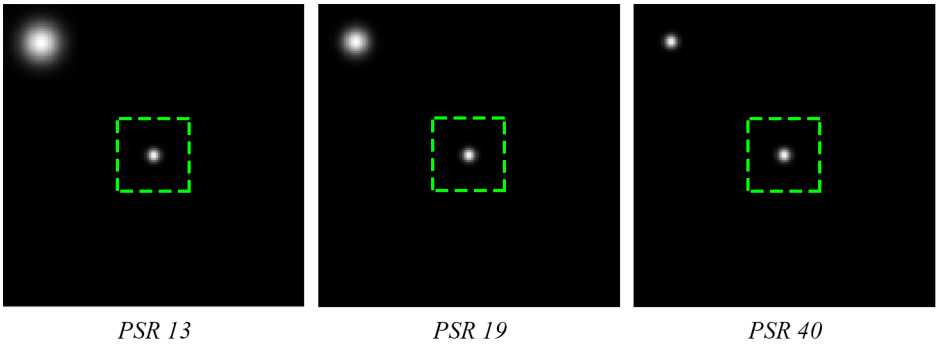


Fig. 4.14 The green box represents a 32 $\times$ 32 window around the maximum point. The remaining area represents the sidelobe in which a region of Gaussian noise has been added to represent a second peak. The larger the amount of noise, the lower the PSR value becomes and the less sure we can be that we have found the correct target.

Using a mean-shift approximation to obtain the change in image coordinates $(\Delta\hat{x}, \Delta\hat{y})$ and assuming the simultaneous shift of all points in the same direction does not include a rotation of the head model, the global shift in the face points can let us adjust the translation parameters since

$$\begin{bmatrix} w(\hat{x}+\Delta\hat{x}) \\ w(y+\Delta y) \\ w \end{bmatrix} = \mathbf{K} \begin{bmatrix} t_x+\Delta t_x \\ t_y+\Delta t_y \\ t_z+\Delta t_z \end{bmatrix}$$

$$\begin{bmatrix} w\hat{x} \\ w\hat{y} \\ w \end{bmatrix} + \begin{bmatrix} w\Delta\hat{x} \\ w\Delta\hat{y} \\ 0 \end{bmatrix} = \mathbf{K} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \mathbf{K} \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ \Delta t_z \end{bmatrix} \tag{4.74}$$

therefore

$$\begin{bmatrix} w\Delta\hat{x} \\ w\Delta\hat{y} \\ 0 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \Delta t_x \\ \Delta t_y \\ \Delta t_z \end{bmatrix} \tag{4.75}$$

Since $w = t_z$ and $\Delta t_z = 0$ we have simple updates for the t_x and t_y pose translation parameters

$$\Delta t_x = \frac{t_z \Delta\hat{x}}{f_x} \tag{4.76}$$

$$\Delta t_y = \frac{t_z \Delta\hat{y}}{f_y} \tag{4.77}$$

4.7 Summary

The final fitting process for the 2.5D Constrained Local Model is described in Algorithm (1). This chapter introduced a novel head pose estimator using a 3D shape model and small 2D texture patches around each point in the model. One suitable texture filter (MOSSE) has been heavily optimised to run efficiently on mobile platforms. The model obtains the 3D pose estimation directly through the use of a tailor-made Jacobian that calculates how the shape is deformed after it has been projected into the image with a full-perspective camera model. It has

been built in such a way that the origin of the head pose lies directly between the inner eye corners. In the following chapter a new model for eye-gaze estimation on mobile devices will be introduced that has the same parameters for pose and an identical origin allowing the two models to be perfectly synchronised.

Algorithm 1 2.5D CLM Fitting

Precompute
  Calibrate camera intrinsic matrix $\mathbf{K}$ (eqn. 4.6)
  Construct 3D mean shape $\mathbf{s}_0$ and deformable eigenvectors $\mathbf{\Phi}$ (eqn. 4.9)
  Train 2D texture filters $\mathcal{H}_k$ around n PDM points (e.g. eqn. 4.34)
End

procedure FIT(deformation $\mathbf{d}$, pose $\mathbf{p}$, image $\mathcal{I}$)
  Use $\mathbf{p}$ to crop, rotate and scale $\mathcal{I}$, isolate head image $\mathcal{I}_{face}$ (sec. 4.5.2)
  if First frame or tracking lost **then**
    Detect face using Haar classifier (sec. 4.6.4)
    Build global correlation filter (eqn. 4.34)
  else
    Generate global response map (eqn. 4.35)
    Check for tracking failure (eqn. 4.73)
    Estimate global shift between frames (eqn. 4.76)
    Update global filter (eqn. 4.70)
  repeat
    for $k \leftarrow 1, ..., n$ points in head model **do**
      Generate Γ_k using local regions of $\mathcal{I}_{face}$ and $\mathcal{H}_k$ (e.g. eqn. 4.35)
      Obtain mean-shift estimates $\mathbf{v}_k$ from Γ_k (eqn. 4.50)
      Project the 3D point $\mathbf{X}_k$ into the image space $\mathcal{W}(\mathbf{X}_k, \mathbf{q})$ (eqn. 4.59)
      Calculate deformable parameter derivatives $\frac{\partial \mathbf{X}_k}{\partial \mathbf{d}}$ (eqn. 4.62)
      Evaluate pose derivatives $\frac{\partial \tilde{\mathbf{X}}_k}{\partial \tilde{\mathbf{r}}}$ (eqn. 4.65) and $\frac{\partial \tilde{\mathbf{X}}_k}{\partial \mathbf{t}}$ (eqn. 4.66)
      Evaluate current derivatives of the projection $\frac{\partial \mathcal{W}(\mathbf{X}_k, \mathbf{q})}{\partial \mathbf{q}}$ (eqn. 4.61)
    Combine derivatives to make final Jacobian matrix $\mathbf{J}$ (eqn. 4.57)
    Compute the parameter updates $\Delta\mathbf{q} = \begin{bmatrix} \Delta\mathbf{d} \\ \Delta\mathbf{p} \end{bmatrix}$ (eqn. 4.56)
    Update deformation parameters $\mathbf{d} \leftarrow \mathbf{d} + \Delta\mathbf{d}$
    Update pose $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$, with rotation adjustment (sec. 4.6.3)
    Determine the new 3D points $\mathbf{X}$ (eqn. 4.10)
  until $(\|\Delta\mathbf{q}\| < \epsilon)$ or $(iter = maxIter)$

Chapter 5

Eye-Gaze Estimation on Mobile Devices

While traditional eye-gaze estimation methods utilise a mapping from the eyes to the screen, the mapping is only valid while the user's head remains static to the calibration position. By utilising the head pose calculations from the previous chapter it is possible to derive the eye-gaze of a user toward a monitor plane through geometric means.

5.1 Constrained Geometric Binocular Model

Once the head pose has been estimated appropriately, the question becomes how to incorporate this new knowledge into a gaze estimation strategy. An obvious choice perhaps is to create a geometric estimation whereby given known camera, monitor and eye coordinates a ray vector can be determined which intersects the monitor at the point-of-gaze. In effect this removes the mapping and thus the need to recalibrate after moving the head from its initial position. While geometric methods are underutilised, they are not new (Pirri et al., 2011; Wood and Bulling, 2014). This work seeks to improve upon these methods by analysing each step of the geometric principles in detail and refining the process such that the optimisation method is constrained and tends toward the correct location, increasing robustness and accuracy.

A new model is derived called the Constrained Geometric Binocular Model which takes into account both eyes simultaneously. The output of the model optimisation process is the 2D coordinates (u, v) on the monitor plane, with the rotation of one eye directly affecting the orientation of the other. The model itself is similar to that of the head model and is again a 2.5D model with a 2D texture representation along with a 3D Point Distribution Model (PDM). A further calculation is introduced by asking how the change of parameters within the PDM of each eye affect the final gaze location.

A complex Jacobian is calculated which models each part of the geometric model separately and derives an analytic derivative. An exception to this is in determining how the eye-orientation changes as the parameters of the PDM change since there is no geometric properties that define this. A multivariate manifold is introduced which effectively maps how the eye looks under different spherical rotations. This manifold is not dependent on the current pose of the user and thus remains fixed, allowing for a numerical derivative to be approximated and saved in look-up tables.

The model is complete with a binocular shape model which includes both eyes along with the nose points for stability. The PDM for each eye is determined via thousands of examples from a synthetic dataset and are fixed according to the inner eye corner allowing for easy interaction with the head pose model. The general flow of the algorithm can be seen in Figure (5.1).

For clarity, this chapter covers the following main novel contributions:

- Constrained Geometric Binocular Model (with perspective camera)
- Eye-orientation Manifold
- Binocular shape model creation
- Calculation of the monitor plane Jacobian

5.2 The eye model

The model of the eye is kept relatively simple due to the lack of high resolution images or infra-red lights to examine the eye in more detail. The eye is modelled

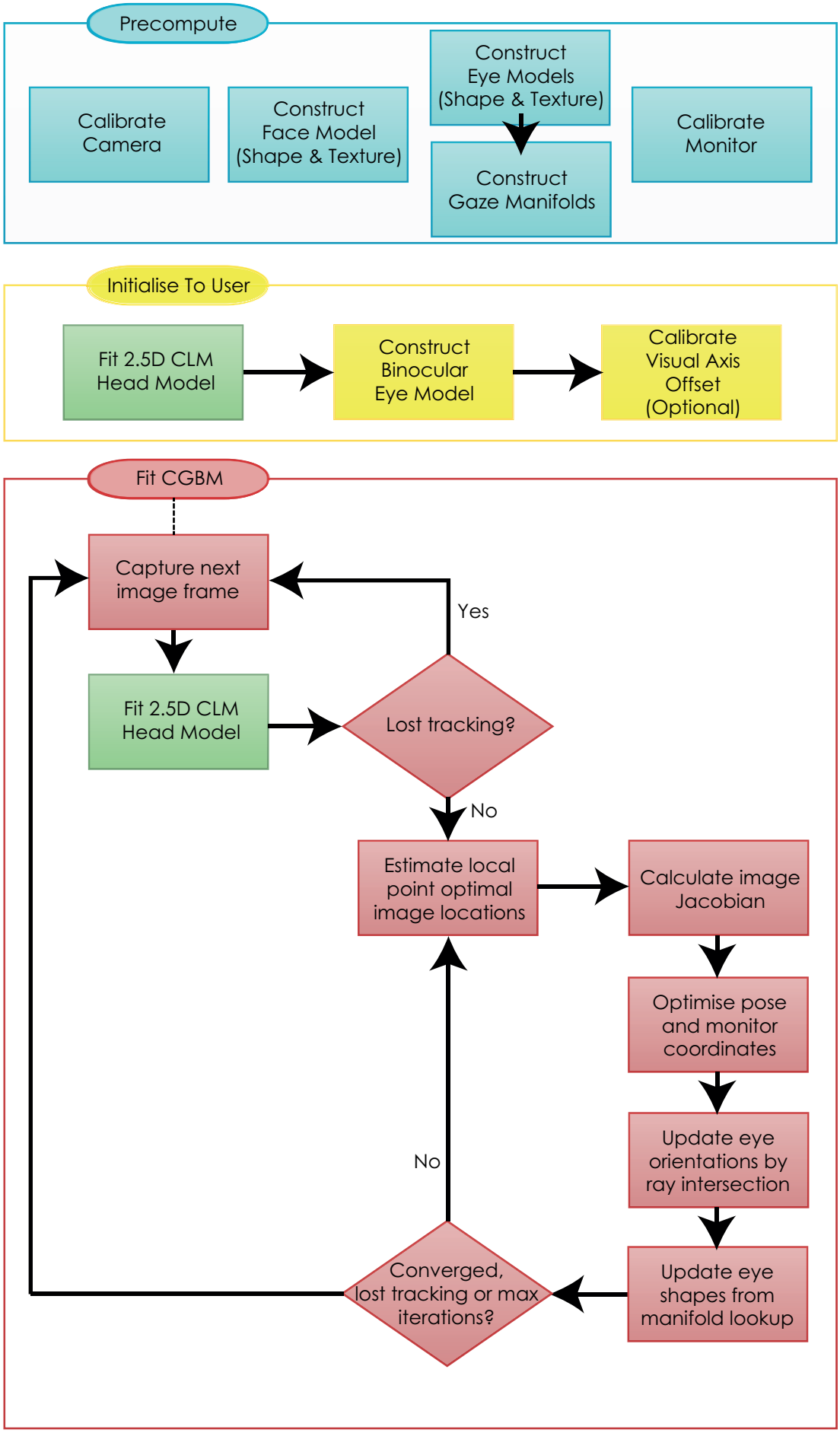


Fig. 5.1 Constrained Geometric Binocular Model flowchart

as a sphere with its centre at some location from the origin of the head model. This is possible due to the fixed origin of the head models located midway between the inner eye corners. The location for each eye is considered fixed for each user and can be adjusted through the initial estimates from the head model. Spherical coordinates are used extensively throughout in the representation of the eye-gaze angles. They help formulate the direction of both the optical axis and also the more complicated calculations for the visual axis i.e. the true line of gaze.

A novelty of this work is that rather than calculate the eye gaze independently for each eye and averaging the result, through geometric means a 'binocular' model is built that optimises the joint visual axis intersection onto the monitor display. This ensures that both eyes help each other toward the optimal point-of-gaze and are kept in sync, although it should be noted that the model will still work if only one eye is available.

5.2.1 The image eye model

Similarly to the head model, within the image itself the eye shape is a 2.5D CLM comprising of a 3D PDM and local patch experts representing the texture. The model represents the 'generic' eye containing a mean shape along with eigenvectors describing how the eye model deforms. Obviously, to obtain the necessary variety in the dataset a large number of tagged images is required along with the 3D measurements. This is not practical to obtain. To overcome this limitation the synthetic eye models generated by the freely available program UnityEyes by (Wood et al., 2016*b*) are utilised. The model has 20 shape points in total, comprising of 8 points evenly distributed around the iris, 5 points each on the upper and lower eyelids and both eye corners. Through a PCA process the 3D shape model can be derived where all shapes are this time aligned with their origin at the inner eye-corner to allow simpler integration with the head model (Figure 5.2). Similarly, through the tagged images that are output by UnityEyes the patch filters can be trained quickly establishing a fully detectable model within the image. To ensure high accuracy, LNF filters from the OpenFace library (Baltrušaitis et al., 2016) are used as described in section 4.5.

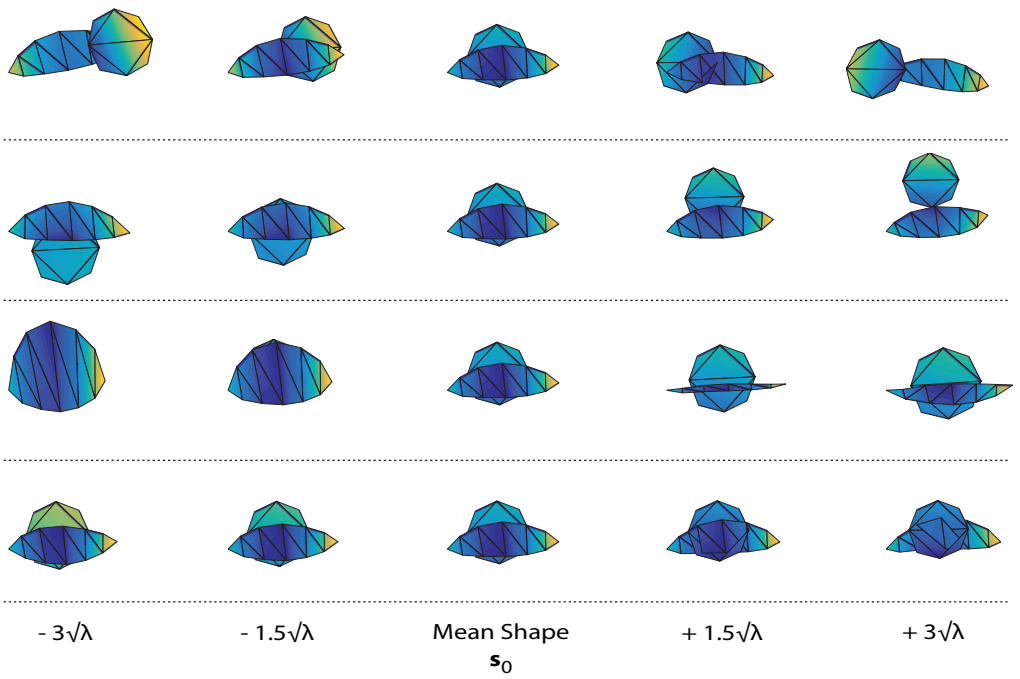


Fig. 5.2 The output for the first four eigenvectors of the left eye. The main variation comes from the movement of the iris relative to the eyelids. The inner eye corner (left hand side of the images) is at the origin at all times with all other points moving around it which allows for easy integration with the head model.

5.2.2 The geometric eye gaze model

In Chapter 2 it was discussed how under the much simpler model of eye gaze; the interpolation based approach; the calibration is often straightforward, for example, via polynomial regression (Cerrolaza et al., 2008). However, the user is required to perform the calibration each time they use the system as it is defined as a mapping between the eye corners and iris to screen coordinates, which is entirely dependent on the pose relationship between camera, head pose and monitor.

For the geometric approach, the model parameters are still dependent on the relationship between the 3 components of the system, but they are modelled explicitly and separately. The eye parameters are actually essentially consistent for the lifetime of a user, only changing slightly over many years (Berrio et al., 2010) and hence only have to be computed in a one-time calibration. The camera-monitor relationship can change but for many mobile solutions with cameras built

in to the device itself (such as most tablets, smartphones, and laptops) a one time calibration is required. Since we have discussed how we calculate the head-camera relationship via pose estimation, we can also calculate the head-monitor relationship and hence determine a user's eye gaze estimation on the monitor plane.

As discussed in Chapter 4 the camera is considered to be at the origin $(0, 0, 0)$ in world space and the head pose is determined in reference to it. To complete the model we need to calibrate the monitor location in world space (i.e. in reference to the camera) which will allow us to determine the relationship between the head pose and the monitor.

5.3 Monitor calibration

In most contexts, the monitor display can be assumed to be a rectangle lying on a plane with limits designated by the 4 corners of the display. Correctly calibrating the monitor is a difficult process since the standard method of estimating pose from a camera is to observe the object within the camera image. Of course under most eye-gaze tracking scenarios the display will never be within the image frame since the camera will be pointing at the user. In most modern laptops, phones and tablets the location of the camera is fixed within the plane of the display itself and the specifications of the device can provide very good estimates for the 4 corners although if the camera is rotated slightly small errors in the estimate are likely. In many cases however the monitor plane can be assumed to have a unit normal of $(0, 0, 1)$, that is pointing directly down the positive *z-axis*.

In cases where a separate camera is attached to a device, either explicit measurements need to be taken carefully or a specific calibration will need to take place involving a mirror as shown by Takahashi et al. (2012). The method involves taking many images from different angles similar to Figure (5.3) whereby the standard camera calibration image is placed onto the display. In practice it can be difficult to capture enough angles for a good estimation and mirror distortion also plays a part. An alternative using a depth camera stream to reduce the error was described in section 3.2.1 on the DMUHAG dataset.

Since the monitor is a plane, we define it in terms of a single 3D point $\mathbf{P}_0$

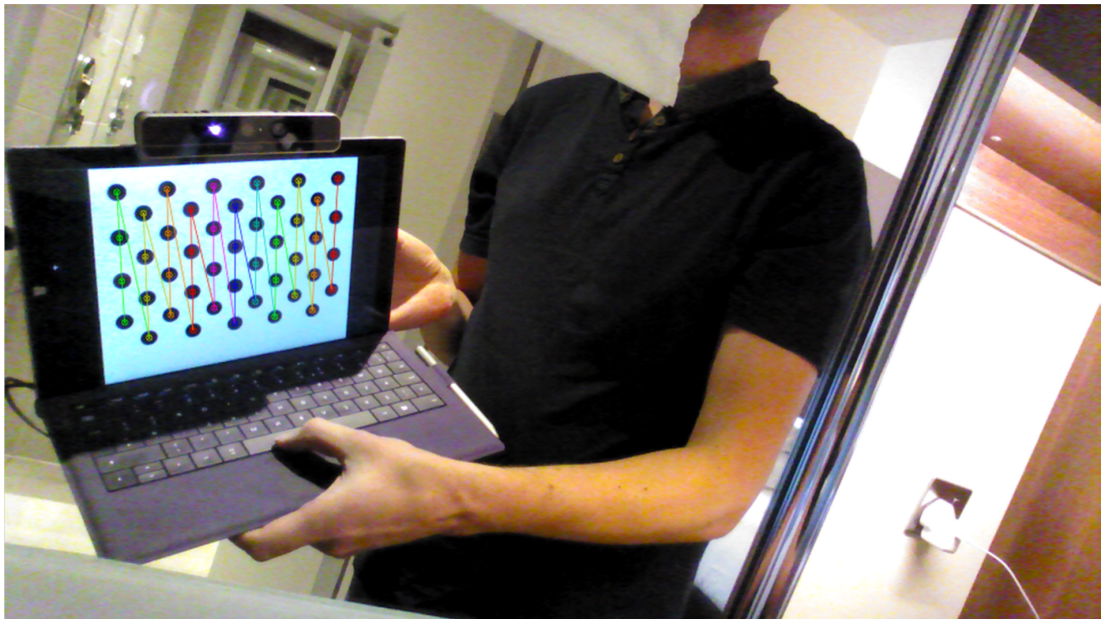


Fig. 5.3 Monitor Calibration through the use of a mirror allowing the monitor to be observable from within the camera image frame

and the plane unit normal $\hat{\mathbf{n}}$. It is useful to have a standard set of 'Monitor Coordinates' which have their origin $(0, 0)$ at the centre of the monitor with the monitor edges $= \{-1, +1\}$ in the u and v axis as shown in Figure (5.4). This provides a standardised way of interacting with software and differing screen resolutions.

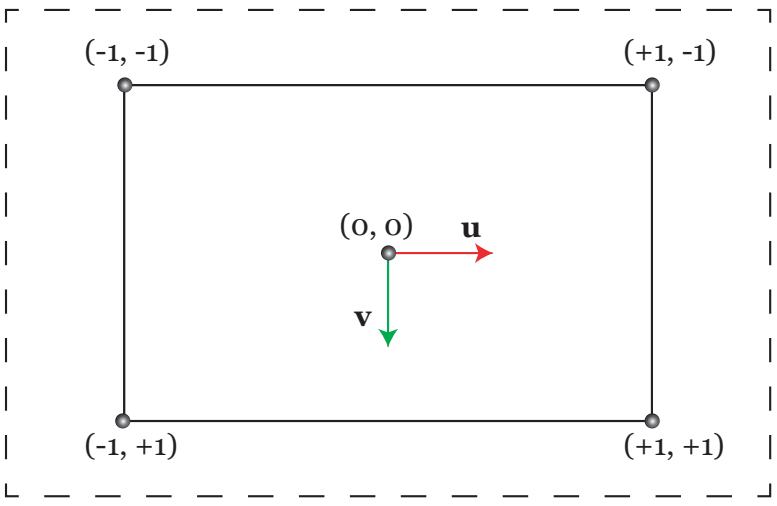


Fig. 5.4 A 2D representation of the monitor. This representation is 'normalised' such that the boundaries lie on -1 and $+1$ allowing for simple migration between displays of varying size and resolution

Denoting the 4 corners of the monitor in $\mathbb{R}^3$ as $\mathbf{P}_{TL}, \mathbf{P}_{TR}, \mathbf{P}_{BL}, \mathbf{P}_{BR}$, we let

$$\mathbf{P}_0 = \frac{\mathbf{P}_{TL} + \mathbf{P}_{BR}}{2} \tag{5.1}$$

be the centre of the monitor screen and

$$\begin{aligned} \mathbf{M}_H &= \frac{\mathbf{P}_{TR} - \mathbf{P}_{TL}}{2} \\ \mathbf{M}_V &= \frac{\mathbf{P}_{BL} - \mathbf{P}_{TL}}{2} \end{aligned} \tag{5.2}$$

represent the horizontal and vertical axes on the display, as seen in Figure (5.5).

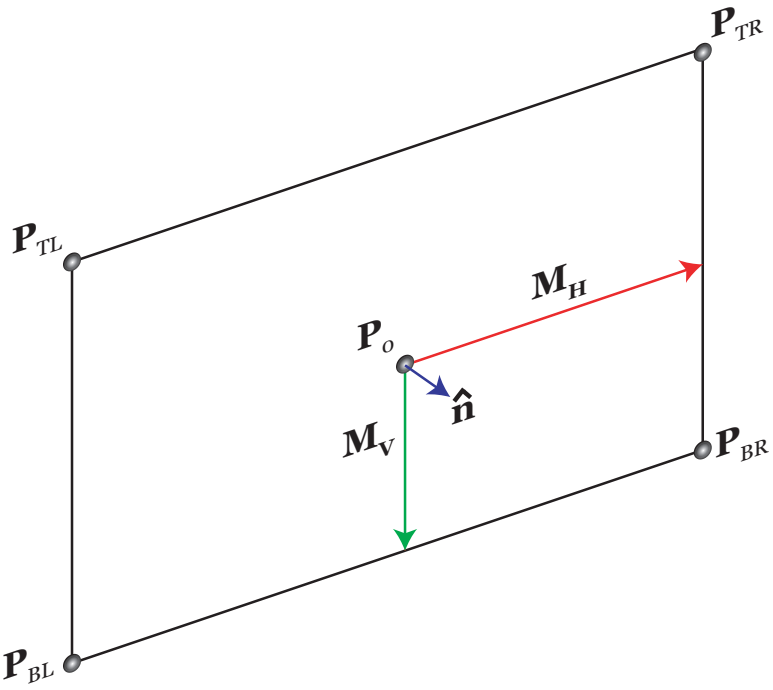


Fig. 5.5 A 3D representation of the monitor display. The monitor is calibrated when the 3D position of the corners are known along with the unit normal. These values are relative to the camera position which in this work defines the 'world' space

A point lying on the monitor plane $\mathbf{Q} \in \mathbb{R}^3$ can be scalar projected onto the monitor axes using the dot product giving the horizontal and vertical distances

from the centre on the screen

$$\begin{aligned} Dist_H &= \frac{\mathbf{M}_H \bullet (\mathbf{Q} - \mathbf{P}_0)}{\|\mathbf{M}_H\|} \\ Dist_V &= \frac{\mathbf{M}_V \bullet (\mathbf{Q} - \mathbf{P}_0)}{\|\mathbf{M}_V\|} \end{aligned} \tag{5.3}$$

Normalising to monitor space, the point $\mathbf{S} \in \mathbb{R}^2$ in the monitor space has coordinates

$$\mathbf{S} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{M}_H \bullet (\mathbf{Q} - \mathbf{P}_0)}{\|\mathbf{M}_H\|^2} \\ \frac{\mathbf{M}_V \bullet (\mathbf{Q} - \mathbf{P}_0)}{\|\mathbf{M}_V\|^2} \end{bmatrix} \tag{5.4}$$

5.4 Eyeball rotation parameters

As previously mentioned, the default pose of the eyeball is to face directly down the positive *z-axis*. Since the eyeball is modelled as a sphere the rotation of the eyeball can be modelled using only 2 parameters under spherical coordinates (where using the optical axis the eyeball radius is considered to be 1 since it has no effect on the direction vector). The angles represent the azimuth angle θ (looking left and right) and the elevation angle ϕ (looking up and down). The model used in this work does not explicitly take into account cyclorotation φ of the eye (torsional movement about the line-of-sight) due to muscle contractions around the eye. However it is implicitly included since Donder's law (Hansard and Horaud, 2010) states the torsion is determined by the gaze direction and so can be treated as a function $\varphi(\theta, \phi)$.

Under spherical coordinates the order of rotation is not important since

$$\mathbf{R}(\theta)\mathbf{R}(\phi) = \mathbf{R}(\phi)\mathbf{R}(\theta) \tag{5.5}$$

and hence the optical line-of-sight $\boldsymbol{\ell}_{opt}$ from the eyeball centre can be computed with a rotation of the unit vector in z about the *y-axis* (azimuth) multiplied by

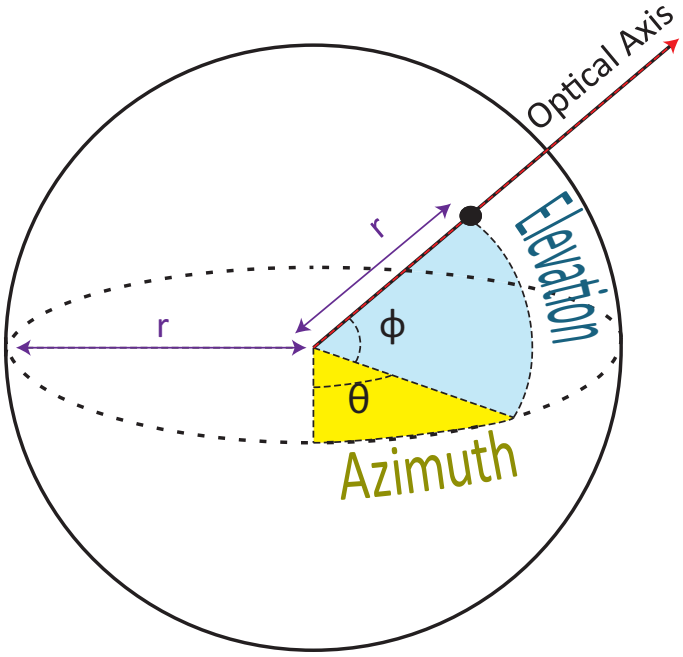


Fig. 5.6 Spherical coordinates of the eye

a rotation about the *x-axis* (elevation) to give

$$\boldsymbol{\ell}_{opt} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} -\sin(\theta)\cos(\phi) \\ \sin(\phi) \\ -\cos(\theta)\cos(\phi) \end{bmatrix} \tag{5.6}$$

5.4.1 Visual axis calculations

It is also possible to describe the more complicated approximation of the eye gaze along the visual axis in a similar way. Under rotation, the nodal point $\mathbf{n}$ is a distance d away from the eyeball centre and positioned at

$$\mathbf{n} = d \begin{bmatrix} -\sin(\theta)\cos(\phi) \\ \sin(\phi) \\ -\cos(\theta)\cos(\phi) \end{bmatrix} \tag{5.7}$$

Let us assume the angle κ can instead be described through components of azimuth (κ_θ) and elevation (κ_ϕ) via a second set of spherical coordinates around the position of the nodal point $\mathbf{n}$ (the new origin of gaze).

$$\boldsymbol{\ell}_{vis\ about\ \mathbf{n}} = \begin{bmatrix} -\sin(\kappa_\theta)\cos(\kappa_\phi) \\ \sin(\kappa_\phi) \\ -\cos(\kappa_\theta)\cos(\kappa_\phi) \end{bmatrix} \tag{5.8}$$

There is no straightforward calculation to combine the two sets of spherical coordinates as their origins are at different places, and their radial distances differ. Hence we seek a new representation of the $\boldsymbol{\kappa}$ angles that instead describe a rotation about the eyeball centre so that they can simply be added to the actual gaze rotation $\boldsymbol{\Theta}$. We denote these new angles $(\hat{\kappa}_\theta, \hat{\kappa}_\phi)$.

Through an alternative representation the normalised direction vector of the visual line-of-gaze can be determined by

$$\boldsymbol{\ell}_{vis} = \frac{\mathbf{p}_\kappa - \mathbf{n}}{t} \tag{5.9}$$

where, as shown in Figure (5.7), $\mathbf{p}_\kappa$ is the 3D point on the eyeball surface passed through by the visual line-of-gaze (at radial distance r from the eyeball centre), and $\mathbf{n}$ is the 3D point representing the position of the nodal point (at radial distance d from the eyeball centre). Note that t is the fixed distance between the

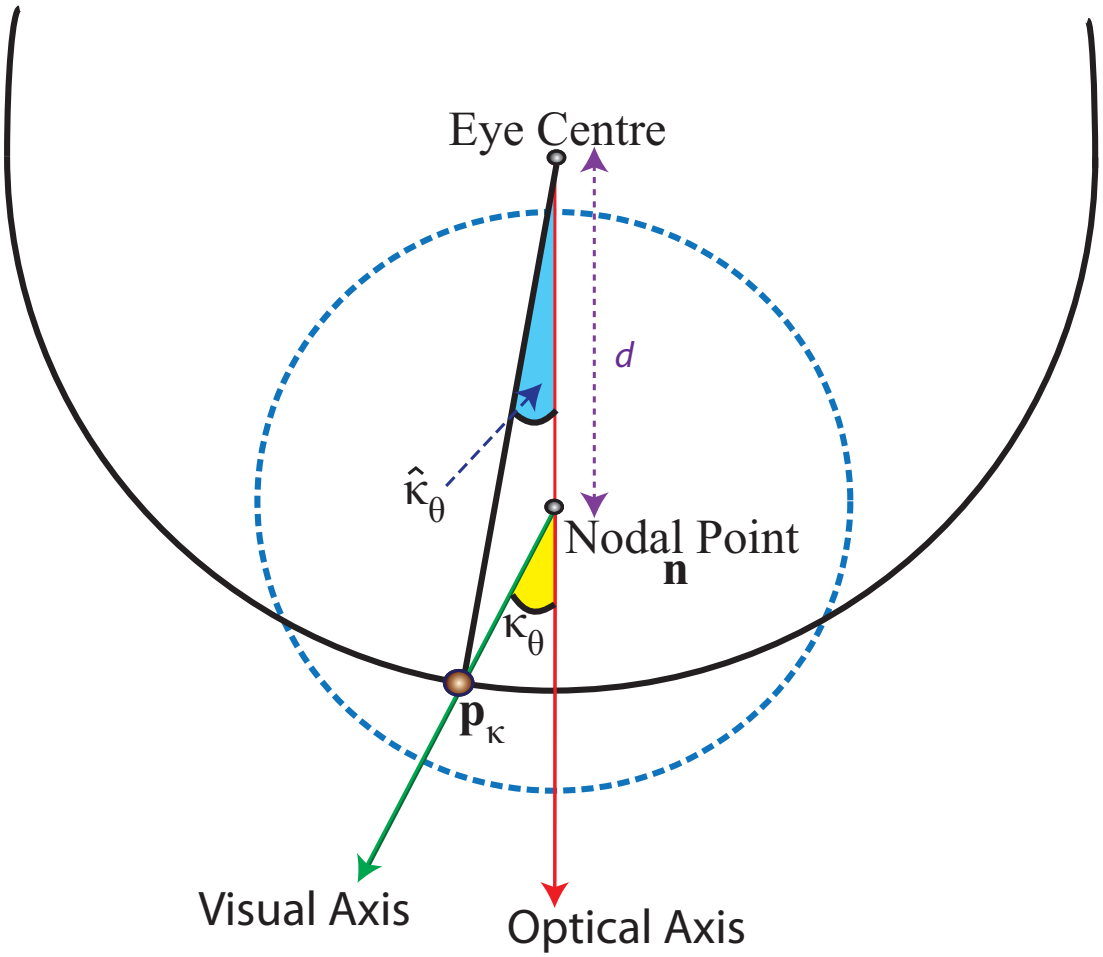


Fig. 5.7 The visual axis passes through the eye sphere at a point $\mathbf{p}_\kappa$. For illustration purposes a cross-section of the eye is used and defines only $\hat{\kappa}_\theta$, but since the spherical coordinates are independent we can obtain $\hat{\kappa}_\phi$ without loss of generality

two points. This allows us to define the point $\mathbf{p}_\kappa$ as

$$\mathbf{p}_\kappa = r \begin{bmatrix} -\sin(\theta + \hat{\kappa}_\theta)\cos(\phi + \hat{\kappa}_\phi) \\ \sin(\phi + \hat{\kappa}_\phi) \\ -\cos(\theta + \hat{\kappa}_\theta)\cos(\phi + \hat{\kappa}_\phi) \end{bmatrix} \tag{5.10}$$

Since the angles $(\hat{\kappa}_\theta, \hat{\kappa}_\phi)$ are fixed for each individual they can be precalculated by observing that when the eyeball is in its default position $(\theta = 0, \phi = 0)$ the visual-axis passes through a point $\mathbf{p}^0_\kappa = \left[\kappa^0_x, \kappa^0_y, \kappa^0_z\right]^\top$ on the eyeball surface. Since we are working from the frame of reference of the eyeball (that is the eyeball origin

is located at $[0,0,0]^\top$, we have

$$\left\|\mathbf{p}_\kappa^0\right\|^2 = r^2 \tag{5.11}$$

From equation (5.9) we know the visual line-of-gaze in its parametric format (with scalar t)

$$\mathbf{p}_\kappa^0 = \begin{bmatrix} \kappa_x^0 \\ \kappa_y^0 \\ \kappa_z^0 \end{bmatrix} = \mathbf{n}^0 + t\ {\boldsymbol{\ell}_{vis}}^0 \tag{5.12}$$

which when combined with equation (5.11) gives

$$\left\|\mathbf{n}^0 + t\ {\boldsymbol{\ell}_{vis}}^0\right\|^2 = r^2 \tag{5.13}$$

Expanding the equation gives

$$\left\|\mathbf{n}^0\right\|^2 + 2t\left({\boldsymbol{\ell}_{vis}}^0 \bullet \mathbf{n}^0\right) + t^2\left\|{\boldsymbol{\ell}_{vis}}^0\right\|^2 = r^2 \tag{5.14}$$

Since by construction the visual direction is a unit vector, $\left\|{\boldsymbol{\ell}_{vis}}^0\right\|^2 = 1$. Futhermore $\left\|\mathbf{n}^0\right\|^2 = d^2$ which gives the following quadratic equation

$$t^2 + 2t\left({\boldsymbol{\ell}_{vis}}^0 \bullet \mathbf{n}\right) + (d^2 - r^2) = 0 \tag{5.15}$$

which can be solved for $t > 0$ using the quadratic formula

$$t = -\left({\boldsymbol{\ell}_{vis}}^0 \bullet \mathbf{n}^0\right) \pm \sqrt{\left({\boldsymbol{\ell}_{vis}}^0 \bullet \mathbf{n}^0\right)^2 + (r^2 - d^2)} \tag{5.16}$$

Finally, rotating about the eyeball centre gives us the representation

$$\mathbf{p}_{\kappa}^{0} = r \begin{bmatrix} -\sin(\hat{\kappa}_{\theta})\cos(\hat{\kappa}_{\phi}) \\ \sin(\hat{\kappa}_{\phi}) \\ -\cos(\hat{\kappa}_{\theta})\cos(\hat{\kappa}_{\phi}) \end{bmatrix} \tag{5.17}$$

which when combined with the evaluated point $\mathbf{p}_{\kappa}^{0} = \left[\kappa_x^0, \kappa_y^0, \kappa_z^0\right]^{\top}$ from equation (5.12) gives

$$\hat{\kappa}_{\phi} = \arcsin\left(\frac{\kappa_y^0}{r}\right) \tag{5.18}$$

$$\hat{\kappa}_{\theta} = \arcsin\left(\frac{-\kappa_x^0}{r\cos(\hat{\kappa}_{\phi})}\right) \tag{5.19}$$

The final visual line-of-gaze can be determined by its source $\boldsymbol{\ell}_{\mathbf{0}vis} = \mathbf{n}$ and its direction vector

$$\boldsymbol{\ell}_{vis} = r \begin{bmatrix} -\sin(\theta + \hat{\kappa}_{\theta})\cos(\phi + \hat{\kappa}_{\phi}) \\ \sin(\phi + \hat{\kappa}_{\phi}) \\ -\cos(\theta + \hat{\kappa}_{\theta})\cos(\phi + \hat{\kappa}_{\phi}) \end{bmatrix} - d \begin{bmatrix} -\sin(\theta)\cos(\phi) \\ \sin(\phi) \\ -\cos(\theta)\cos(\phi) \end{bmatrix} \tag{5.20}$$

5.4.2 Line-of-gaze monitor intersection

Whether utilising the visual or optical axis, it is important to convert them into world coordinates in order to determine their intersection with the monitor plane. To obtain the world coordinates of the source point $\boldsymbol{\ell}_0^w$ and the direction $\boldsymbol{\ell}^w$ it is required to utilise the binocular model pose $(\mathbf{R}|\mathbf{t})$.

$$\boldsymbol{\ell}_0^w = \mathbf{R}\boldsymbol{\ell}_0 + \mathbf{t} \tag{5.21}$$

$$\boldsymbol{\ell}^w = \mathbf{R}\boldsymbol{\ell} \tag{5.22}$$

The geometric properties of the eye are now defined as an origin $\boldsymbol{\ell}_o^w$ and a

direction vector $\boldsymbol{\ell}^w$ giving a 'ray' in world coordinates. As discussed in section (5.3) the monitor is defined as a plane so our problem is to determine the point $\mathbf{r}_i \in \mathbb{R}^3$ where the ray intersects the plane.

The ray $\mathbf{r}$ is defined between its origin and infinity. This gives us a line parameterised by a scalar t where $0 \leq t \leq \infty$

$$\mathbf{r} = \boldsymbol{\ell}_o^w + t\, \boldsymbol{\ell}^w \tag{5.23}$$

Provided the direction vector $\boldsymbol{\ell}^w$ is not parallel with the plane, a positive value of t can be found when the vector intersects the plane (a negative value occurs when the gaze direction is away from the plane). Since our plane is defined as a point $\mathbf{P}_0$ and a unit normal $\hat{\mathbf{n}}$, the ray intersection $\mathbf{r}_i$ occurs when

$$(\mathbf{P}_0 - \mathbf{r}_i) \bullet \hat{\mathbf{n}} = 0 \tag{5.24}$$

Combining equations (5.23) and (5.24) allows us to solve for the parameter t on intersection

$$t = \frac{(\mathbf{P}_0 - \boldsymbol{\ell}_0^w) \bullet \hat{\mathbf{n}}}{\boldsymbol{\ell}^w \bullet \hat{\mathbf{n}}} \tag{5.25}$$

Creating a final ray intersection equation

$$\mathbf{r}_i = \boldsymbol{\ell}_o^w + \frac{(\mathbf{P}_0 - \boldsymbol{\ell}_0^w) \bullet \hat{\mathbf{n}}}{\boldsymbol{\ell}^w \bullet \hat{\mathbf{n}}}\, \boldsymbol{\ell}^w \tag{5.26}$$

5.5 A binocular shape model

Rather than calculate the parameters of each eye individually it is better to utilise a model that contains both eyes. This is because since the eye can look very similar over a wide variety of differing angles relative to the camera, allowing free movement of the eye tends to find only local minima. In fact even with both eyes calculated in harmony, there can be little difference between the eyes fully tilted downward versus the eyelid being nearly closed. To counter this a new model

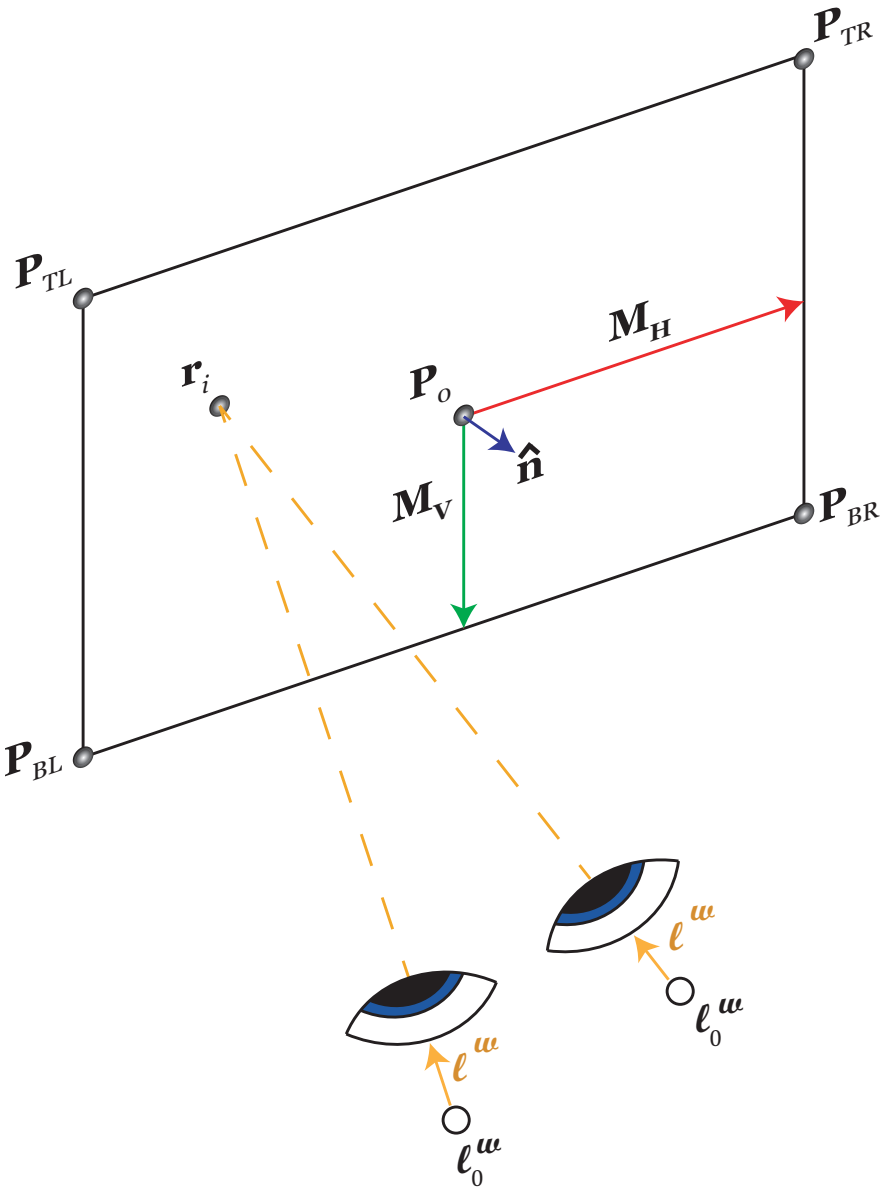


Fig. 5.8 Each eye has its own ℓ_0^w (the source of the gaze vector) and ℓ^w (the direction of gaze). These world vectors determine the gaze from each eye as a ray which crosses the monitor plane at an intersection point $\mathbf{r}_i$, assumed in this work to be identical for both.

is built that contains anchor points that do not deform and are only subject to pose changes. The nose points make an excellent choice as anchor points since after they have been initialised to the individual's face they are mostly static with very little ability to move and as such there is no further requirement for them to deform. The initialisation of the anchor points are taken directly from the initial head pose estimate allowing for continued refinement of the pose as well as the eye parameters.

Since the eye models were purposely generated such that the inner eye corners do not move through parameter adjustment, they also make for good anchor points. As shown in Figure (5.9), the new model comprises 49 points (20 on

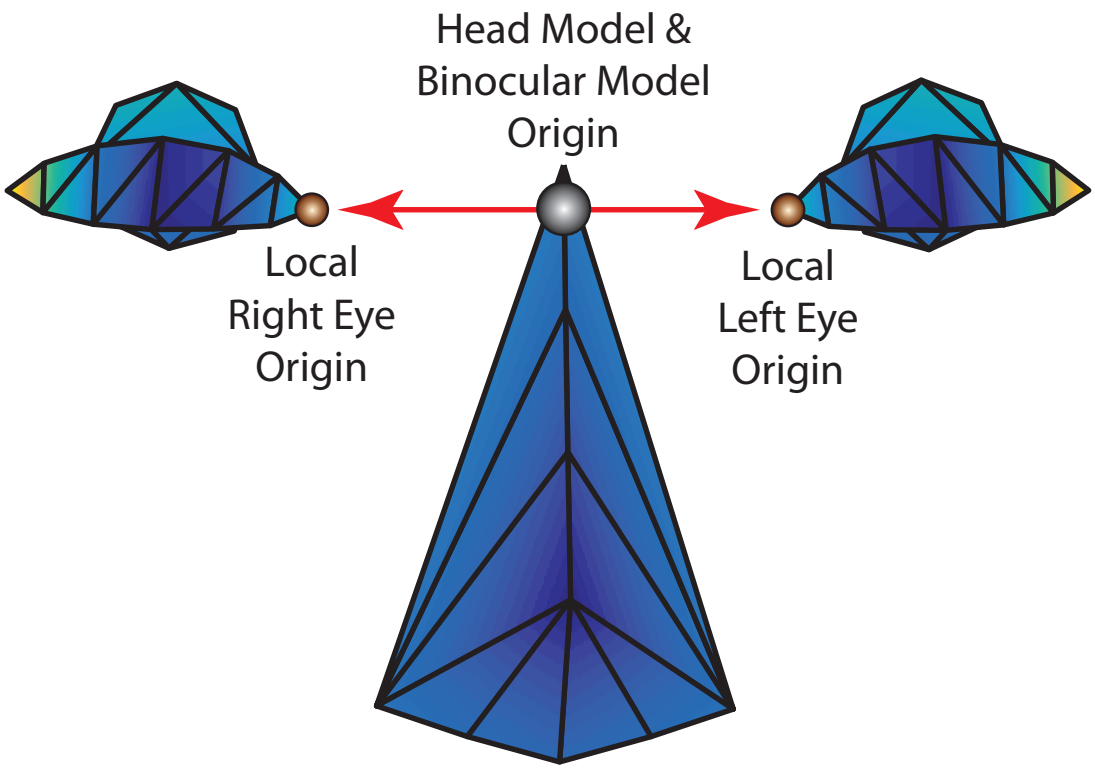


Fig. 5.9 The binocular model combines two eye PDMs together with the points on the nose which remain fixed in place on construction. The origin of the new model is identical to that of the head model allowing for easy interoperability. The inner eye corners remain along the local *x-axis* but can be shifted 'symmetrically' as required from the initial output of the head model. The inner eye corners act as their respective eye's local origin and therefore by design they too are fixed in place. It therefore becomes a straightforward process to reconcile the deformation parameters of each eye with the new model.

each eye and 9 from the nose). During a calibration procedure of the user, a purpose built shape model is generated which uses the average locations of the nose points and inner eye corners from the general model and fixes them in place making them permanently rigid components. The left and right eye models are duplicated at the inner eye corner locations to complete the model. This gives the mean shape of the model $\mathbf{s}_0^{binoc}$ as

$$\mathbf{s}_0^{binoc} = \begin{bmatrix} Reye_{p1} & \dots & Reye_{p20} & Leye_{p1} & \dots & Leye_{p20} & Nose_{p1} & \dots & Nose_{p9} \end{bmatrix}^\top \tag{5.27}$$

The model has 20 eigenvectors (10 for each eye). At this point, the eyes remain independent of one another and can rotate such that each eye is not restricted to 'look' at any individual point-of-gaze. In other words, the eigenvectors $\mathbf{\Phi}^{binoc}$ become

$$\mathbf{\Phi}^{binoc} = \begin{bmatrix} \begin{bmatrix} d_1^{Reye_{p1}} & \dots & d_{10}^{Reye_{p1}} \\ \vdots & \vdots & \vdots \\ d_1^{Reye_{p20}} & \dots & d_{10}^{Reye_{p20}} \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} d_1^{Leye_{p1}} & \dots & d_{10}^{Leye_{p1}} \\ \vdots & \vdots & \vdots \\ d_1^{Leye_{p20}} & \dots & d_{10}^{Leye_{p20}} \end{bmatrix} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{5.28}$$

In this guise the binocular model is constrained such that both eyes and nose work together for the pose parameters, but the deformable parameters do not work under the same constraints. It is possible to undertake the same optimisation as described in Chapter 4 to determine the parameters of the eye and from there estimate the line-of-gaze from each eye independently through estimating the iris normal from the calculated points.

5.5.1 Constraining the deformable parameters

A goal of this work is to further improve the model by providing an extra constraint of the deformable parameters of the eye, that is to say, both eyes must be fixed upon or very close to the same position within 3D world space. In reality, since the eyes can process a region which becomes less detailed as it approaches the peripheral, obtaining a specific optimum location can be tricky. Using both eyes together helps to refine the optimum gaze estimation which suits both eyes.

In the majority of cases where we wish to determine eye-gaze from a non-head-mounted device, the optimum location we are after lies on the monitor plane and so we are really looking to solve an equation which looks for the optimum screen coordinates $\mathbf{S} = (u, v) \in \mathbb{R}^2$ that both eyes are looking toward. Adding these to the 6 pose parameters as before gives us 8 gaze parameters $\mathbf{g}$ to optimally solve for.

Suppose given the 8 parameters suggested above it is possible to generate a projection within the image of the 49 points of the binocular model described in section (5.5).

$$\mathcal{P}(\mathbf{g}) = \begin{bmatrix} \mathcal{W}(\mathbf{X}_1, \mathbf{g}) \\ \mathcal{W}(\mathbf{X}_2, \mathbf{g}) \\ \vdots \\ \mathcal{W}(\mathbf{X}_n, \mathbf{g}) \end{bmatrix} \tag{5.29}$$

Given the patch experts for each point $\mathbf{x}$ on the eye, a set of optimal locations $\mathbf{y}$ within a new incoming image $\mathcal{I}$ can be determined, with the goal being to find the gaze parameters which best fit these optimal locations. The difference between the current projection and the optimal is our *data* term

$$\mathcal{D}(\mathbf{x}, \mathcal{I}) = \|\mathbf{y} - \mathcal{P}(\mathbf{g})\|^2 \tag{5.30}$$

Note that again there exists some prior knowledge about the problem which can be exploited to assist the fitting process. Firstly, since we already have a

good fit for the pose from the head tracker, there is only a need to make small adjustments. Secondly, the gaze is assumed to be toward the display and so we wish to penalise gaze vectors that significantly deviate from it. Similarly to equation (4.25), the *regularisation* term is defined

$$\mathcal{R}(\mathbf{g}) = r \left\| \mathbf{g} - \mathbf{g}_H \right\|^2_{\mathbf{\Lambda}^{-1}} \tag{5.31}$$

where $\mathbf{g}_H = [r_x, r_y, r_z, t_x, t_y, t_z, 0, 0]^\top$ i.e. the final pose result of the head pose tracker and r is a regularisation weight of our choosing. The squared Mahalanobis distance is determined from the $\mathbf{\Lambda}$ term, which contains suitably acceptable variances for the parameters. Empirically, values of 9, 0.01 and 0.25 for transformation, rotation and screen coordinates respectively produced satisfactory results. The value r is again set to the mean of the variances.

Thus, regularised appropriately, we are looking to find the parameters $\hat{\mathbf{g}}$ that minimise the distance between these optimal locations and the projected points

$$\begin{aligned} \hat{\mathbf{g}} &= \underset{\mathbf{g}}{\operatorname{argmin}} \left\{ \mathcal{R}(\mathbf{g}) + \mathcal{D}(\mathbf{x}, \mathcal{I}) \right\} \\ &= \underset{\mathbf{g}}{\operatorname{argmin}} \left\{ r \left\| \mathbf{g} - \mathbf{g}_H \right\|^2_{\mathbf{\Lambda}^{-1}} + \left\| \mathbf{y} - \mathcal{P}(\mathbf{g}) \right\|^2 \right\} \end{aligned} \tag{5.32}$$

As in Chapter 4, we make use of a first order Taylor series expansion of the projected landmarks

$$\mathcal{P}(\mathbf{g} + \mathbf{\Delta g}) \approx \mathcal{P}(\mathbf{g}) + \mathbf{J}\mathbf{\Delta g} \tag{5.33}$$

where $\mathbf{J}$ is the Jacobian matrix containing the derivatives of the projected shape points $\mathcal{P}(\mathbf{g})$ with respect to the gaze parameters $\mathbf{g}$ i.e.

$$\mathbf{J} = \frac{\partial \mathcal{P}(\mathbf{g})}{\partial \mathbf{g}} \tag{5.34}$$

The concatenated mean-shift vectors $\mathbf{v}$ again represent the optimal movement

projections for all the points within the image frame

$$\mathbf{v} = \mathbf{y} - \mathcal{P}(\mathbf{g}) \tag{5.35}$$

allowing us to solve iteratively using Gauss-Newton optimisation.

$$\mathbf{\Delta g} = \left(r\mathbf{\Lambda}^{-1} + \mathbf{J}^T\mathbf{J}\right)^{-1}\left(\mathbf{J}^T\mathbf{v} - r\mathbf{\Lambda}^{-1}\left(\mathbf{g} - \mathbf{g}_H\right)\right) \tag{5.36}$$

Section (5.4.2) described the relationship between the spherical coordinates of each eye and the monitor screen. What remains missing then is the relationship between the point distribution model and the spherical coordinates.

5.5.2 Manifold learning of the PDM parameters to spherical coordinates

The general PDM of the eye covers a ν-dimensional space by design and captures the wide variety of possible eye shapes and positions for a large number of people. Optimising for the correct eye gaze of a user involves searching this space for the parameters which minimise the observed landmarks within the image. However, for a specific user many of the regions within this space do not represent a valid eye shape and thus the applicable area can be assumed to cover a subspace with $d < m$ dimensions.

A manifold is a topological space that approximates Euclidean space around each local point, similar to how a 2D atlas approximates the 3D surface of the Earth. It is possible then to simplify our model further by suggesting that all of the possible points representing all possible rotations of the user's eye lie on or near a 2D manifold, as we have already shown the eye rotation to depend upon spherical coordinates $\mathbf{\Theta} = (\theta, \phi)$.

The manifold is defined as

$$\mathcal{M}(\mathbf{\Theta}) = \begin{bmatrix} d_1(\mathbf{\Theta}) \\ \vdots \\ d_\nu(\mathbf{\Theta}) \end{bmatrix} \tag{5.37}$$

which effectively represents a function with parameters $\mathbf{\Theta}$ that returns a feature vector of the deformable parameters $\mathbf{d}$. It is important to choose a suitable regression method from the PDM parameters to the $\mathbf{\Theta}$ parameters on the manifold. Since the manifold is a limited subspace ranging from approximately $\theta = \pm 32°, \phi = \pm 32°$ (a reasonable assumption for comfortable eye rotations), it is computationally tractable to collect enough samples to cover a large area of the whole manifold in a short period of time in order to perform a non-parametric regression.

Since the eigenvectors themselves were created via PCA they are orthogonal to one another and thus can be assumed to be independent of each other. As such the problem becomes ν separate regression problems to the two-dimensional manifold $\boldsymbol{\theta}$. Lu et al. (2014) showed that a 2D manifold corresponding to coordinates on a monitor screen from a user with fixed head pose is locally linear and perform an adaptive linear regression with sparsely collected data in order to not be obtrusive to the user during calibration. Since the simulated eye model is capable of generating millions of samples with noise only incurred from the creation of the simulation itself, it is straightforward to cover the whole domain with samples and utilise a smoothing process. This ensures the manifolds are continuous and smooth and the differentials can be numerically calculated with reasonable precision.

The manifold itself is sampled at 0.5° intervals and stored as a 128×128 discrete grid structure for each ν. A non-parametric method of regression is used to create the 2D mapping from the samples as shown in Figure (5.10). This is a Nadaraya-Watson Kernel Regression (Nadaraya, 1964; Watson, 1964) where the output can be approximated at each location by a locally weighted sum

$$\mathcal{M}(\mathbf{\Theta}) = \frac{\sum_{i=1}^{n} \mathcal{K}_h(\mathbf{\Theta} - \mathbf{\Theta}_i)\, \mathbf{d}_i}{\sum_{i=1}^{n} \mathcal{K}_h(\mathbf{\Theta} - \mathbf{\Theta}_i)} \tag{5.38}$$

Fig. 5.10 The manifold $\mathcal{M}(\boldsymbol{\Theta})$ shows how the ν deformable parameters vary depending on the current θ and ϕ values (colour contrast maximised from lowest values (red) to highest values (blue) for visualisation). Note how through Kernel Regression the manifolds become very smooth and suitable for numerical differentiation.

where $\mathcal{K}_h$ is a Kernel with bandwidth h. The Kernel used is an isotropic Gaussian with $\sigma = 5$ that spans 6° across θ and ϕ in the manifold.

5.6 Jacobian calculation

The Jacobian matrix ($\mathbf{J}$) from equation (5.34) is the derivative of the projected points in the shape model with respect to the gaze parameters $\mathbf{g}$. It defines how a small change in each of the parameters is reflected in a change of the image coordinates. Since the projection is non-linear, the Jacobian matrix is only locally applicable and needs to be calculated each frame. The 8 gaze parameters are made up of the 6 pose parameters $\mathbf{p}$ and the 2 screen parameters $\mathbf{S} = (u, v)$, and thus the Jacobian becomes

$$\begin{aligned}\mathbf{J} &= \begin{bmatrix} \frac{\partial \mathcal{P}(\mathbf{g})}{\partial \mathbf{p}} & \frac{\partial \mathcal{P}(\mathbf{g})}{\partial \mathbf{S}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial \mathcal{P}(\mathbf{g})}{\partial p_1} & \frac{\partial \mathcal{P}(\mathbf{g})}{\partial p_2} & \cdots & \frac{\partial \mathcal{P}(\mathbf{g})}{\partial p_6} & \frac{\partial \mathcal{P}(\mathbf{g})}{\partial u} & \frac{\partial \mathcal{P}(\mathbf{g})}{\partial v} \end{bmatrix}\end{aligned} \tag{5.39}$$

where

$$\frac{\partial \mathcal{P}(\mathbf{g})}{\partial p_i} = \begin{bmatrix} \frac{\partial \mathcal{W}(\mathbf{X}_1, \mathbf{g})}{\partial p_i} \\ \frac{\partial \mathcal{W}(\mathbf{X}_2, \mathbf{g})}{\partial p_i} \\ \vdots \\ \frac{\partial \mathcal{W}(\mathbf{X}_n, \mathbf{g})}{\partial p_i} \end{bmatrix}, \quad \frac{\partial \mathcal{P}(\mathbf{g})}{\partial u} = \begin{bmatrix} \frac{\partial \mathcal{W}(\mathbf{X}_1, \mathbf{g})}{\partial u} \\ \frac{\partial \mathcal{W}(\mathbf{X}_2, \mathbf{g})}{\partial u} \\ \vdots \\ \frac{\partial \mathcal{W}(\mathbf{X}_n, \mathbf{g})}{\partial u} \end{bmatrix}, \quad \frac{\partial \mathcal{P}(\mathbf{g})}{\partial v} = \begin{bmatrix} \frac{\partial \mathcal{W}(\mathbf{X}_1, \mathbf{g})}{\partial v} \\ \frac{\partial \mathcal{W}(\mathbf{X}_2, \mathbf{g})}{\partial v} \\ \vdots \\ \frac{\partial \mathcal{W}(\mathbf{X}_n, \mathbf{g})}{\partial v} \end{bmatrix} \tag{5.40}$$

Using the eye-gaze manifold, the calculation to warp the 3D point to the image coordinates becomes

$$\begin{aligned}\mathcal{W}(\mathbf{X}_k, \mathbf{g}) &= \begin{bmatrix}\hat{x}_k \\ \hat{y}_k\end{bmatrix} = E\left(\begin{bmatrix}w_k\hat{x}_k \\ w_k\hat{y}_k \\ w_k\end{bmatrix}\right) = E\left(\mathbf{K}\begin{bmatrix}x_k \\ y_k \\ z_k\end{bmatrix}\right) \\ &= E\left(\begin{bmatrix}f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1\end{bmatrix}\left(\mathbf{R}\begin{bmatrix}\mathbf{s}_0^{x_k} + \mathcal{M}(\mathbf{\Theta})\mathbf{\Phi}^{x_k} \\ \mathbf{s}_0^{y_k} + \mathcal{M}(\mathbf{\Theta})\mathbf{\Phi}^{y_k} \\ \mathbf{s}_0^{z_k} + \mathcal{M}(\mathbf{\Theta})\mathbf{\Phi}^{z_k}\end{bmatrix} + \begin{bmatrix}t_x \\ t_y \\ t_z\end{bmatrix}\right)\right)\end{aligned} \tag{5.41}$$

where again the E operator converts the 3D homogeneous coordinates to 2D Euclidean coordinates (equation 4.5).

In chapter 4 the Jacobian of the projection using a full projection camera model was derived (section 4.6.2). It allowed us to calculate how each point of our model moved within the image by first calculating how the parameters of our model affect the 3D point coordinates. A similar process can be followed here

$$\begin{aligned}\frac{\partial \mathcal{W}(\mathbf{X}_k, \mathbf{g})}{\partial g_i} &= \begin{bmatrix}f_x \frac{\partial}{\partial g_i}\left(\frac{x_k}{z_k}\right) \\ f_y \frac{\partial}{\partial g_i}\left(\frac{y_k}{z_k}\right)\end{bmatrix} \\ &= \frac{1}{z_k^2}\begin{bmatrix}f_x\left(\frac{\partial(x_k)}{\partial g_i}z_k - x_k\frac{\partial(z_k)}{\partial g_i}\right) \\ f_y\left(\frac{\partial(y_k)}{\partial g_i}z_k - y_k\frac{\partial(z_k)}{\partial g_i}\right)\end{bmatrix}\end{aligned} \tag{5.42}$$

and hence with respect to each parameter in $\mathbf{g}$ we simply need to find the derivative of the 3D point and substitute it into equation (5.42).

The k^{th} 3D point of the shape $\mathbf{X}_k = [x_k, y_k, z_k]^\top$ with parameters $\mathbf{g}$ is defined

as

$$\mathbf{X}_k(\mathbf{g}) = \mathbf{R}\begin{bmatrix} \mathbf{s}_0^{x_k} + \mathcal{M}(\boldsymbol{\Theta})\boldsymbol{\Phi}^{x_k} \\ \mathbf{s}_0^{y_k} + \mathcal{M}(\boldsymbol{\Theta})\boldsymbol{\Phi}^{y_k} \\ \mathbf{s}_0^{z_k} + \mathcal{M}(\boldsymbol{\Theta})\boldsymbol{\Phi}^{z_k} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{5.43}$$

The first parameters of the Jacobian are based on the pose changes of the 3D model which has already been explored and remains as in equations (4.65) and (4.66). The remaining 2 parameters are the screen coordinates $\mathbf{S} = (u, v)$ and so we need to define the derivative of the 3D points of the binocular shape model with respect to those parameters. i.e. $\frac{\partial \mathbf{X}}{\partial \mathbf{S}}$.

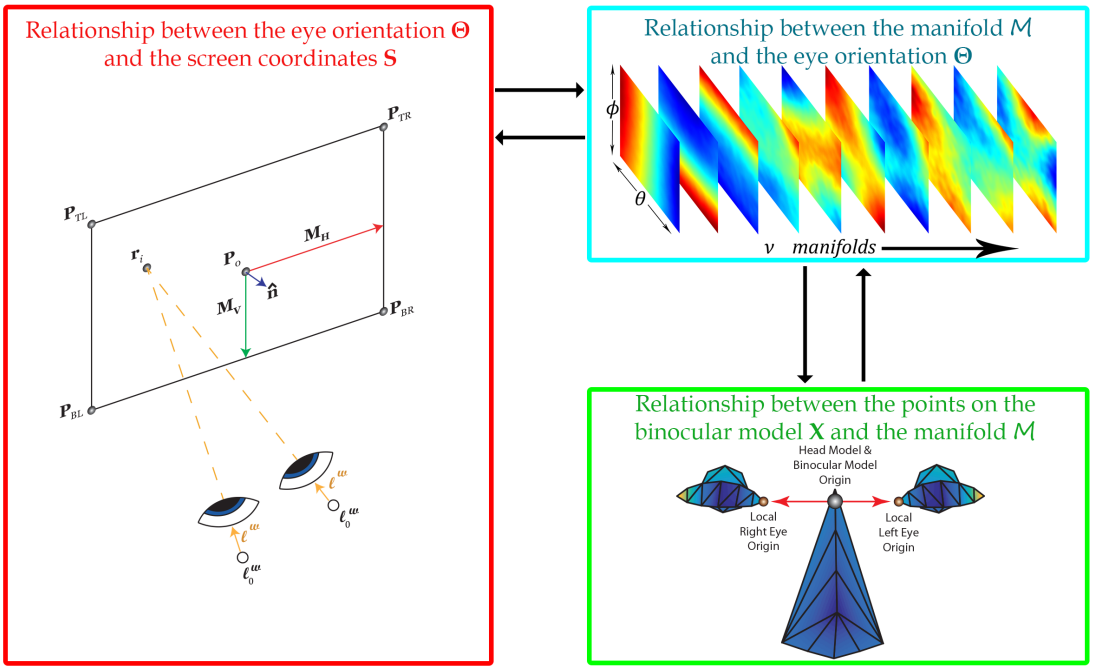


Fig. 5.11 Flow from the binocular model to the screen. The Jacobian will help describe how a change in the gaze coordinate on the screen changes the shape of the binocular model and its calculation can be broken down into its separate parts using the chain rule.

As shown in Figure (5.11), the geometric design of the work creates a sequence of formulas that depend on one another. The three constituent parts are bound by the colours red, blue and green that will be used for clarity in the following section. The screen coordinates **S** are dependent on its 3D monitor coordinate

$\mathbf{r}_i$, which is determined by the four spherical coordinates (two from each eye $\boldsymbol{\Theta}$) representing the direction of the eye-gaze. By design these in turn are dependent on the manifold $\mathcal{M}$ and the parameters $\mathbf{d}$ on that manifold. The parameters $\mathbf{d}$ then give us the shape of the model through the eigenvectors $\boldsymbol{\Phi}$. While the overall derivative can seem complicated, each can be computed in turn and combined at the end. In this way some of the components are precomputed after a training stage and can quickly be determined from a lookup table. Utilising the chain rule we have

$$\frac{\partial \mathbf{X}}{\partial \mathbf{S}} = \frac{\partial \mathbf{X}}{\partial \mathcal{M}(\boldsymbol{\Theta})} \quad \frac{\partial \mathcal{M}(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} \quad \frac{\partial \boldsymbol{\Theta}}{\partial \mathbf{S}} \tag{5.44}$$

5.6.1 Derivatives of the first two components

The first two components of $\frac{\partial \mathbf{X}}{\partial \mathbf{S}}$ for each point are individually quite straightforward. Firstly, taking the derivative of $\mathbf{X}_k$ in equation (5.43) with respect to the manifold $\mathcal{M}(\boldsymbol{\Theta})$ gives

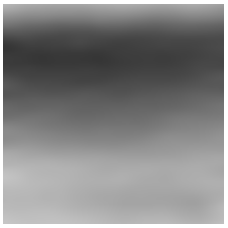
$$\frac{\partial \mathbf{X}_k}{\partial \mathcal{M}(\boldsymbol{\Theta})} = \begin{bmatrix} \frac{\partial (x_k)}{\partial \mathcal{M}(\boldsymbol{\Theta})} \\ \frac{\partial (y_k)}{\partial \mathcal{M}(\boldsymbol{\Theta})} \\ \frac{\partial (z_k)}{\partial \mathcal{M}(\boldsymbol{\Theta})} \end{bmatrix} = \mathbf{R} \begin{bmatrix} \boldsymbol{\Phi}_i^{x_k} \\ \boldsymbol{\Phi}_i^{y_k} \\ \boldsymbol{\Phi}_i^{z_k} \end{bmatrix} \tag{5.45}$$

which is simply the relevant eigenvectors rotated to match the model rotation. Next, taking the derivative of the manifold from equation (5.37) with respect to $\boldsymbol{\Theta}$ gives

$$\frac{\partial \mathcal{M}(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} = \begin{bmatrix} \frac{d_1(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} \\ \vdots \\ \frac{d_\nu(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} \end{bmatrix} \tag{5.46}$$

which can be numerically calculated directly on creation of the manifold maps by convolving an image gradient kernel over them (Figure 5.12). In this work a 3×3 Scharr gradient operator is utilised over the maps, horizontally for the θ

(a) $\frac{d_2(\boldsymbol{\Theta})}{\partial\theta}$



(b) $\frac{d_2(\boldsymbol{\Theta})}{\partial\phi}$

Fig. 5.12 Derivatives of the second manifold parameter w.r.t. θ and ϕ respectively.

derivative and vertically for ϕ

$$\frac{d_n(\boldsymbol{\Theta})}{\partial\theta} = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix} * \frac{d_n(\boldsymbol{\Theta})}{32}$$

$$\frac{d_n(\boldsymbol{\Theta})}{\partial\phi} = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix} * \frac{d_n(\boldsymbol{\Theta})}{32} \tag{5.47}$$

where $*$ is a 2-dimensional convolution operator.

5.6.2 Derivatives of eye-orientation w.r.t. the screen

The derivatives of $\boldsymbol{\Theta}$ with respect to the screen coordinates $\mathbf{S}$ is more tricky and is itself best broken down into multiple parts (coloured for clarity). The way to approach it is to work backwards and invert the final result

$$\frac{\partial\boldsymbol{\Theta}}{\partial\mathbf{S}} = \left(\frac{\partial\mathbf{S}}{\partial\mathbf{r}_i} \quad \frac{\partial\mathbf{r}_i}{\partial\boldsymbol{\ell}^w} \quad \frac{\partial\boldsymbol{\ell}^w}{\partial\boldsymbol{\Theta}}\right)^{-1} \tag{5.48}$$

The calculated derivative is a 2×2 matrix which is easily inverted provided its determinant is non-zero.

The derivative of any 2D screen coordinate $\mathbf{S}$ in equation (5.4) with respect to its 3D location $\mathbf{Q}$ is by construction dependent on the 3D monitor horizontal and vertical vectors, which are constant since the screen position is fixed relative to the camera. Therefore for any intersection point $\mathbf{r}_i = \mathbf{Q}$ the derivative is

$$\frac{\partial \mathbf{S}}{\partial \mathbf{r}_i} = \begin{bmatrix} \frac{M_H^X}{\|\mathbf{M}_H\|^2} & \frac{M_H^Y}{\|\mathbf{M}_H\|^2} & \frac{M_H^Z}{\|\mathbf{M}_H\|^2} \\ \frac{M_V^X}{\|\mathbf{M}_V\|^2} & \frac{M_V^Y}{\|\mathbf{M}_V\|^2} & \frac{M_V^Z}{\|\mathbf{M}_V\|^2} \end{bmatrix} \tag{5.49}$$

The derivative of the ray projection equation (5.26) with respect to the gaze direction vector $\boldsymbol{\ell}^w$ is

$$\frac{\partial \mathbf{r}_i}{\partial \boldsymbol{\ell}^w} = [(\mathbf{P}_0 - \boldsymbol{\ell}_0^w) \bullet \hat{\mathbf{n}}] \frac{\mathbf{I}_3 (\boldsymbol{\ell}^w \bullet \hat{\mathbf{n}}) - \boldsymbol{\ell}^w \hat{\mathbf{n}}^\top}{(\boldsymbol{\ell}^w \bullet \hat{\mathbf{n}})^2} \tag{5.50}$$

where $\mathbf{I}_3$ is a 3×3 identity matrix.

Finally, combining equations (5.22) and (5.6) and taking the derivatives with respect to $\boldsymbol{\Theta} = (\theta, \phi)$ gives

$$\frac{\partial \boldsymbol{\ell}^w}{\partial \boldsymbol{\Theta}} = \begin{bmatrix} \frac{\partial \boldsymbol{\ell}_x^w}{\partial \theta} & \frac{\partial \boldsymbol{\ell}_x^w}{\partial \phi} \\ \frac{\partial \boldsymbol{\ell}_y^w}{\partial \theta} & \frac{\partial \boldsymbol{\ell}_y^w}{\partial \phi} \\ \frac{\partial \boldsymbol{\ell}_z^w}{\partial \theta} & \frac{\partial \boldsymbol{\ell}_z^w}{\partial \phi} \end{bmatrix} = \mathbf{R} \begin{bmatrix} \frac{\partial \boldsymbol{\ell}_x}{\partial \theta} & \frac{\boldsymbol{\ell}_x}{\partial \phi} \\ \frac{\partial \boldsymbol{\ell}_y}{\partial \theta} & \frac{\boldsymbol{\ell}_y}{\partial \phi} \\ \frac{\partial \boldsymbol{\ell}_z}{\partial \theta} & \frac{\boldsymbol{\ell}_z}{\partial \phi} \end{bmatrix} \tag{5.51}$$

When using the optical axis, this derivative equates to

$$\frac{\partial \boldsymbol{\ell}_{opt}^w}{\partial \boldsymbol{\Theta}} = \mathbf{R} \begin{bmatrix} -\cos(\theta)\cos(\phi), & \sin(\theta)\sin(\phi) \\ 0, & \cos(\phi) \\ \sin(\theta)\cos(\phi), & \cos(\theta)\sin(\phi) \end{bmatrix} \tag{5.52}$$

and for the visual axis

$$\frac{\partial \boldsymbol{\ell}_{vis}^{w}}{\partial \boldsymbol{\Theta}} = \mathbf{R}\left(r\begin{bmatrix} -\cos(\theta+\hat{\kappa}_\theta)\cos(\phi+\hat{\kappa}_\phi), & \sin(\theta+\hat{\kappa}_\theta)\sin(\phi+\hat{\kappa}_\phi) \\ 0, & \cos(\phi+\hat{\kappa}_\phi) \\ \sin(\theta+\hat{\kappa}_\theta)\cos(\phi+\hat{\kappa}_\phi), & \cos(\theta+\hat{\kappa}_\theta)\sin(\phi+\hat{\kappa}_\phi) \end{bmatrix} - d\begin{bmatrix} -\cos(\theta)\cos(\phi), & \sin(\theta)\sin(\phi) \\ 0, & \cos(\phi) \\ \sin(\theta)\cos(\phi), & \cos(\theta)\sin(\phi) \end{bmatrix}\right) \tag{5.53}$$

5.7 The UV update

On obtaining the new gaze update parameters $\mathbf{\Delta g}$ from equation (5.36), the model details have to be updated. The manner of updating the translation and rotation parameters is identical to the head pose algorithm and details can be found in section (4.6.3). This in turn updates the positions of the eye centres. For the 2 remaining parameters a simple addition can take place to find the updated 2D monitor coordinates $(u, v) \leftarrow (u, v) + (\Delta u, \Delta v)$.

We seek to find the correct eye PDM parameters within the manifold that correspond to the correct eye rotations, and therefore need to follow a process which goes from the (u, v) coordinates $\rightarrow$ (r_i) 3D Monitor intersection point $\rightarrow$ (θ, ϕ) Spherical Coordinates for each eye $\rightarrow$ manifold parameters for each eye.

The 3D coordinate of the estimated intersection point is

$$\mathbf{r_i} = \mathbf{P}_0 + u\mathbf{M}_H + v\mathbf{M}_V \tag{5.54}$$

with $\mathbf{P}_0$ again being the centre of the monitor screen and $\mathbf{M}_H$ and $\mathbf{M}_V$ the horizontal and vertical axes on the display.

For the optical axis model of gaze, determining a unit normal corresponding

to the direction of gaze is straightforward. Since $\boldsymbol{\ell}_o^w$ is simply the centre of the eye,

$$\boldsymbol{\ell}^w = \frac{\mathbf{r_i} - \boldsymbol{\ell}_o^w}{\|\mathbf{r_i} - \boldsymbol{\ell}_o^w\|} \tag{5.55}$$

which can be rotated back to local coordinates

$$\boldsymbol{\ell} = \mathbf{R}^T \boldsymbol{\ell}^w \tag{5.56}$$

This leads to a simple derivation of the spherical coordinates

$$\phi_{opt} = \arcsin\left(\boldsymbol{\ell}_y\right) \tag{5.57}$$

$$\theta_{opt} = \arcsin\left(\frac{-\boldsymbol{\ell}_x}{\cos(\phi_{opt})}\right) \tag{5.58}$$

5.7.1 Updating from the visual axis

When utilising the visual axis of gaze, calculating the spherical coordinates is somewhat more tricky since the current nodal point position is unknown (as it moves under eye rotation). Note the situation in Figure (5.13). The triangle formed from the vectors denoting the nodal point, optical axis and visual axis has scalar lengths d, a and b respectively. Additionally the internal angles are denoted D, A and B and add up to π radians.

Fortunately, the length d is constant (being the fixed distance the nodal point is from the eyeball centre) and once we have determined the κ point $\mathbf{p}_\kappa^0$ on the eyeball surface, the angle A remains constant by definition and is equal to

$$A = arccos\left(\frac{-\mathbf{n}^0 \bullet \boldsymbol{\ell}_{vis}{}^0}{d}\right) \tag{5.59}$$

At runtime, the length $a = \|\mathbf{r}_i - \mathbf{eyeCentre}\|$, and through the Sine rule the other sides and angles can all be determined:

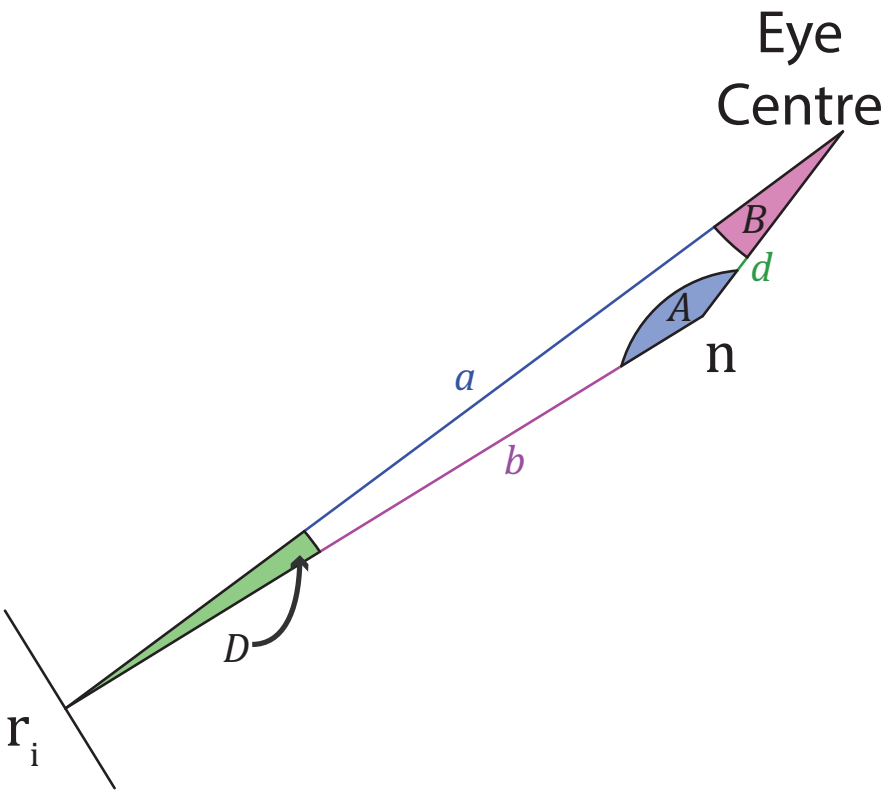


Fig. 5.13 The triangle formed in $\mathbb{R}^3$ by the eye centre, nodal point and monitor intersection $\mathbf{r}_i$. The scalar lengths and angles can all be determined to help obtain the visual axis spherical coordinates.

$$D = \arcsin\left(\frac{d\sin(A)}{a}\right) \tag{5.60}$$

$$B = \pi - A - D \tag{5.61}$$

$$b = \frac{a\sin(B)}{\sin(A)} \tag{5.62}$$

Now the scalar distance b between the nodal point and monitor intersection point is known we can construct a vector $\mathbf{r}_i^0$ that is the same distance away from the eye centre as the monitor intersection point when the eye is in its unoriented pose (Figure 5.14). Since this point lies on the same sphere as $\mathbf{r}_i$, their spherical coordinates can be added and subtracted without issue. As shown in Figure (5.15), the spherical coordinates of $\mathbf{r}_i^0$ can be denoted θ^0 and ϕ^0 and act as the spherical coordinate deviations between the visual and optical axes estimations.

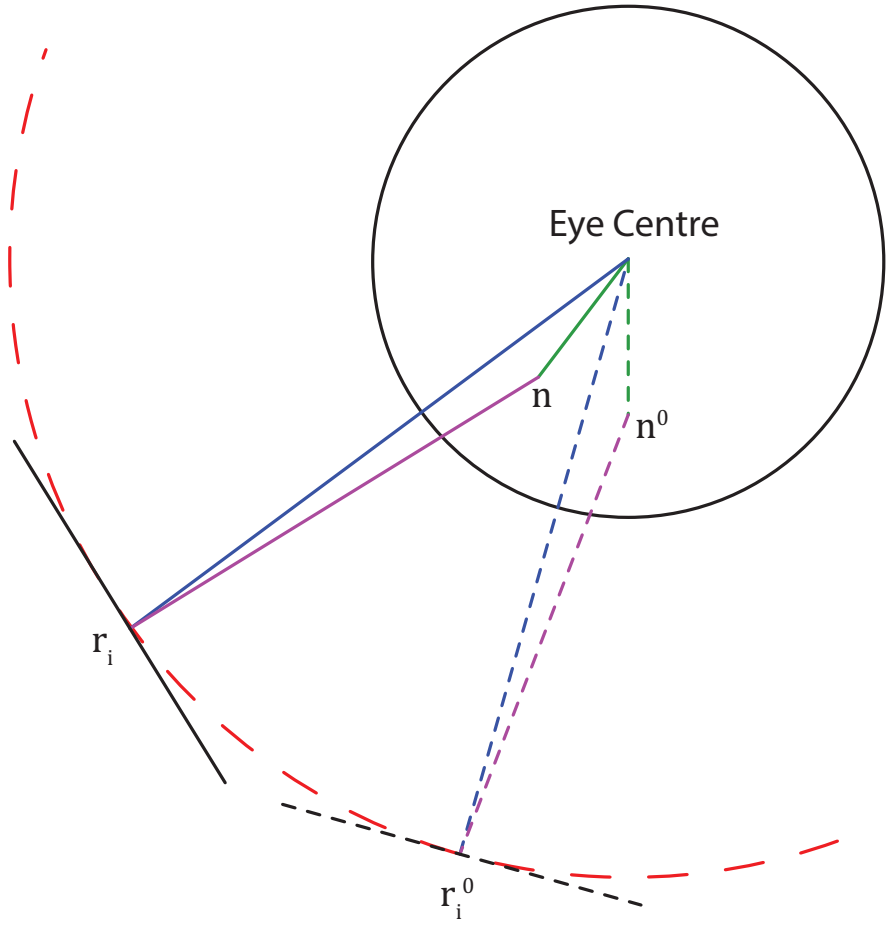


Fig. 5.14 Construction of the point $\mathbf{r}_i^0$ which represents the gaze intersection if θ and ϕ were 0. Its usefulness comes from the fact that it lies on the same spherical surface as $\mathbf{r}_i$.

Note that these are not constant and fluctuate depending only on the distance of the observed object from the eye centre.

Once the spherical coordinates are known, the manifolds themselves then serve as lookup tables which take the two angle coordinates and provide the ν deformation parameters to recreate the eye shape.

5.8 Summary

The final fitting process for the Constrained Geometric Binocular Model is described in Algorithm (2). This chapter introduced the Constrained Geometric

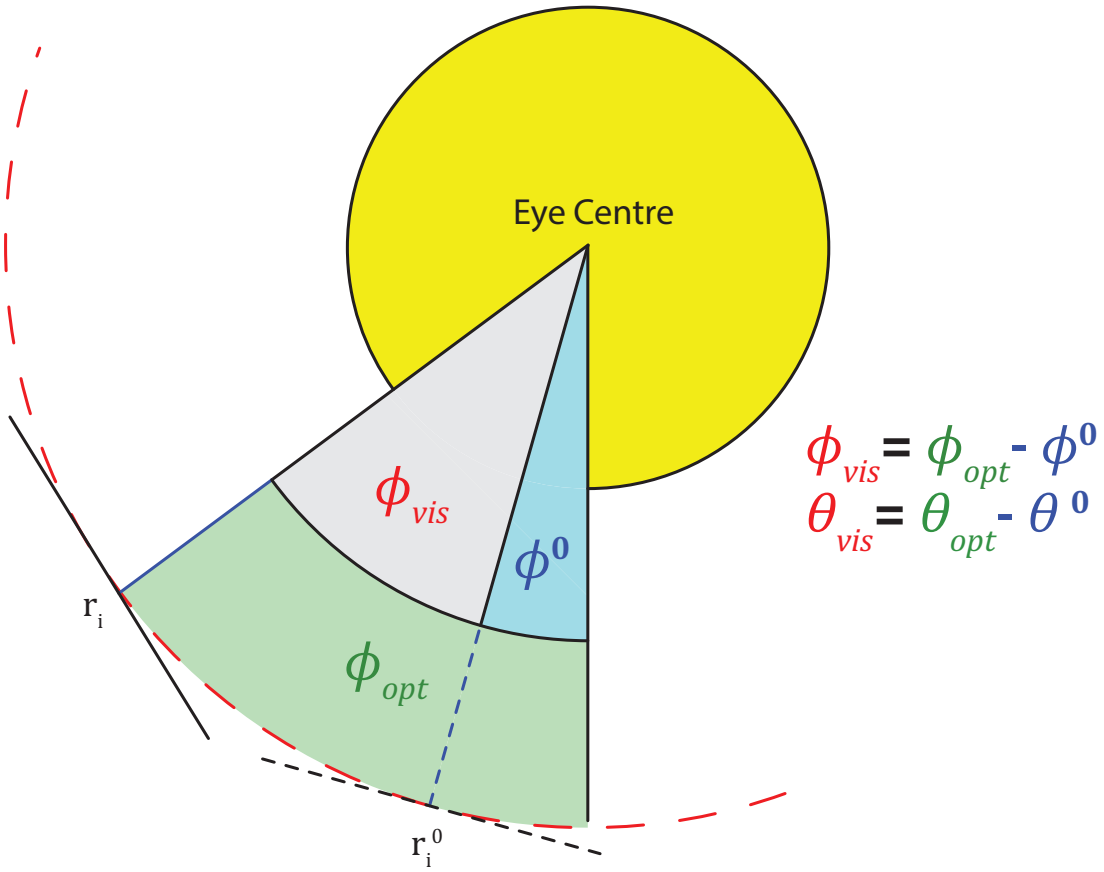


Fig. 5.15 By first obtaining the spherical coordinates for the optical axis, we can simply subtract the offset value ϕ^0 to obtain the visual axis angles. Note this extends to the θ values as well as the coordinates are independent.

Binocular Model - a novel eye-gaze estimator perfectly suited for mobile platforms. The model relies upon and is perfectly synchronised with the head pose estimator seen in the previous chapter. The binocular shape model contains both sets of eye points and is kept anchored by a set of points around the nose. Using a synthetic eye model, an eye-orientation manifold is computed that defines the 3D point distribution at a particular gaze for each eye. The estimated coordinates on a calibrated display screen are then optimised through the use of a complex series of equations and Jacobian matrices that enforce a common PoR for both eyes. In the following chapter, the new head-pose and eye-gaze models will undertake a collection of tests that will determine their accuracy, robustness and suitability for use on mobile devices.

Algorithm 2 Constrained Geometric Binocular Model Fitting

Precompute
  Calibrate camera intrinsic matrix $\mathbf{K}$ (eqn. 4.6)
  Construct mean shape $\mathbf{s}_0$ and deformable eigenvectors $\mathbf{\Phi}$ for each eye (eqn. 4.9)
  Train 2D texture filters around PDM points for each eye (e.g. eqn. 4.34)
  Numerically construct the manifolds $\mathcal{M}(\mathbf{\Theta})$ (eqn. 5.38)
  Numerically construct the manifold derivatives $\frac{\partial \mathcal{M}(\mathbf{\Theta})}{\partial \mathbf{\Theta}}$ (eqn. 5.47)
  Calibrate monitor position and obtain intersection derivative $\frac{\partial \mathbf{S}}{\partial \mathbf{r}_i}$ (eqn. 5.49)
End

Initialise
  Fit 2.5D Constrained Local Model of Head (Algorithm 1)
  Construct binocular model $\mathbf{s}_0^{binoc}$ and deformable eigenvectors $\mathbf{\Phi}^{binoc}$ (sec. 5.5)
  if Using visual axis **then**
    Estimate visual axis κ angle offset (1 point calibration) (sec. 2.2)
End

procedure FIT(deformation $\mathbf{d}$, pose $\mathbf{p}$, image $\mathcal{I}$)
  Fit 2.5D Constrained Local Model of Head (Algorithm 1)
  Use $\mathbf{p}$ to crop, rotate and scale $\mathcal{I}$, isolate eye images $\mathcal{I}_{Eye}$ (sec. 4.5.2)
  repeat
    for *Eye* $\in$ (*leftEye, rightEye*) **do**
      for $k \leftarrow 1, ..., n$ points in eye **do**
        Generate Γ_k using local regions of $\mathcal{I}_{Eye}$ and $\mathcal{H}_k$ (e.g. eqn. 4.32)
        Obtain mean-shift estimates $\mathbf{v}_k$ from Γ_k (eqn. 4.50)
        Project the 3D point $\mathbf{X}_k$ into the image space $\mathcal{W}(\mathbf{X}_k, \mathbf{g})$ (eqn. 5.41)
        Evaluate pose derivatives $\frac{\partial \tilde{\mathbf{x}}_k}{\partial \tilde{\mathbf{r}}}$ (eqn. 4.65) and $\frac{\partial \tilde{\mathbf{x}}_k}{\partial \mathbf{t}}$ (eqn. 4.66)
        Compute the manifold derivatives for the point $\frac{\partial \mathbf{X}_k}{\partial \mathcal{M}(\mathbf{\Theta})}$ (eqn. 5.45)
      Determine eye/monitor derivatives $\frac{\partial \mathbf{r}_i}{\partial \boldsymbol{\ell}^w}$ (eqn. 5.50) and $\frac{\partial \boldsymbol{\ell}^w}{\partial \mathbf{\Theta}}$ (eqn. 5.51)
      Evaluate the gaze differential $\frac{\partial \mathbf{\Theta}}{\partial \mathbf{S}}$ (eqn. 5.48)
    Evaluate current derivatives of the projection $\frac{\partial \mathcal{W}(\mathbf{X}_k, \mathbf{g})}{\partial \mathbf{g}}$ (eqn. 5.42)
    Combine derivatives to make final Jacobian matrix $\mathbf{J}$ (eqn. 5.39)
    Compute the parameter updates $\Delta\mathbf{g} = \begin{bmatrix} \Delta\mathbf{d} \\ \Delta\mathbf{p} \\ \Delta\mathbf{uv} \end{bmatrix}$ (eqn. 5.36)
    Update deformation parameters $\mathbf{d} \leftarrow \mathbf{d} + \Delta\mathbf{d}$
    Update pose $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$, with rotation adjustment (sec. 4.6.3)
    Update the uv parameters $\mathbf{uv} \leftarrow \mathbf{uv} + \Delta\mathbf{uv}$
    Determine the 3D points $\mathbf{X}$ (eqn. 5.43) and ray intersection $\mathbf{r}_i$ (eqn. 5.54)
    For each eye determine the parameters $\boldsymbol{\ell}, \boldsymbol{\ell}_0, \theta, \phi$ (sec. 5.7)
  until $(\|\Delta\mathbf{q}\| < \epsilon)$ or $(iter = maxIter)$

Part IV

Experimental Evaluation

Chapter 6

Results & Discussion

In order to evaluate the proposed new methods for head-pose and eye-gaze estimation, there is a need to construct a number of test cases that reflect typical scenarios that may occur on mobile devices. There is a need to consider a number of issues alongside the accuracy of the system such as its robustness and computational intensity, especially considering that mobile devices typically have restrictions on battery power, memory and CPU cycles. Perhaps more importantly is the need to test on a variety of datasets that cover the three main influencing factors discussed in Chapter 3, that is the varieties of *Camera, Environment* and *User*. The following section presents the results from a series of experiments which attempt to isolate these factors where appropriate while comparing the success and failure of the proposed model to other works seen in the literature where it is available.

6.1 Head pose estimation test results

The first experiment for estimating the head pose accuracy is performed on the UPNA dataset. Since it was performed with a *'flock-of-birds'* tracker and care was taken when calibrating the device, robust ground truth values for both rotation and translation are available. A separate 'model' file was included for each participant where 3D coordinates on the participant (including, crucially, the inner eye corners) are known. The local coordinate system of the supplied ground-truth is

actually located on the top of the participant's head and needs to be transformed through a *constant* pose adjustment to the inner eye corners so that it can be compared correctly. By following the same alignment process that was performed on the creation of the PDM (4.2.3), the supplied tagged point model can be used to determine an estimate of the pose offset which can subsequently be applied on all frames, although there is no guarantee on its accuracy. It is important to stress that this does not in any way affect the impartiality of the proposed tracking model as it remains independent at all times with no additional training taking place.

The 2.5D CLM head pose estimation data was acquired over all video frames in the dataset. No assumptions were made about starting locations, with the initial frame head pose first being estimated by a Haar classifier as outlined in section 4.6.4. All of the videos start with a frontal face and therefore all faces were detected successfully. Table (6.1) compares the new model with the state-of-the-art models tested in Ariz et al. (2016). The authors first estimated the 2D face shape though both an ASM and an AAM which are both commonly seen in the literature. Then the POSIT method (Dementhon and Davis, 1995) was applied which attempts to best fit the 2D points with a known 3D model. Three models of the face were tested – the real 3D data of the participant captured from the *'flock-of-birds'* tracker, a generic 3D head model (the Basel Face Model (Paysan et al., 2009)) and a cylindrical model (CHM).

The 2.5D CLM with LNF texture filters outperforms all other models tested on the mean accuracy of rotation. In addition, it performs better than all the other models on individual rotation angles except by a negligible amount on the yaw parameter of the AAM that was fit with the real head data of the participant. The standard deviation on all three rotation axes is low showing its robust nature. It does not require an explicit 3D model like the POSIT algorithm, instead deforming the original PDM directly to estimate the pose parameters. Thus it is a one-stage process and can therefore be implemented more efficiently than the other methods.

Unfortunately the MOSSE filters, while effective, are seemingly not robust to large changes in yaw and cannot recover as efficiently. As there is no prior on the pose estimation, outliers can have a significant negative effect on the mean. The median errors are summarised in Table (6.2) where it can be seen there is

Method		**Head Rotation Error** (°)			
Tracker	Head Model	Roll	Yaw	Pitch	Mean
ASM & POSIT	Real 3D	0.97	2.47	3.54	2.33 ± 2.48
ASM & POSIT	Basel Face Model	1.12	2.97	4.04	2.71 ± 2.82
ASM & POSIT	Cylindrical Head Model	1.14	3.56	5.52	3.40 ± 3.02
AAM & POSIT	Real 3D	1.52	**1.82**	4.33	2.56 ± 3.94
AAM & POSIT	Basel Face Model	1.74	2.30	6.01	3.35 ± 4.29
AAM & POSIT	Cylindrical Head Model	1.74	3.68	8.83	4.75 ± 4.84
2.5D CLM (MOSSE) ***(this work)***	Point Distribution Model	1.32 ± 3.63	3.81 ± 9.78	3.14 ± 3.77	2.76 ± 6.49
2.5D CLM (LNF) ***(this work)***	**Point Distribution Model**	**0.81 ± 0.59**	1.88 ± 1.18	**2.47 ± 2.09**	**1.72 ± 1.58**

Table 6.1 Mean rotation errors on the UPNA dataset. Mean ± Standard Deviation displayed where known.

no significant difference between the two models. This shows that for the 2.5D CLM generally the errors are consistently low, with larger errors accruing on a small number of occasions, due to minor tracking failure. This kind of tracking failure is very difficult to determine since no 'global' failure has occurred and the facial features are still being tracked. However, since the face is believed to be at a different rotation than it is, the 'wrong' set of patches from the multiple view choices are being used. It is likely that the non-linear nature of the LNF filters provides them a larger tolerance to the variability of the textures from different view angles.

Fig. 6.1 Examples of model fitting on the UPNA dataset. While the fitting is generally good, facial hair can be the source of some inaccuracies as seen in the bottom-right image.

As expected, the pitch angle is the most difficult to determine since most of the frontal face points appear approximately planar to one another with only the position of the tip of the nose in relation to the other points around it providing any strong evidence of pitch angle. One other notable instance of an inaccurate pitch being obtained was when estimating the values of the users with beards, as seen in the bottom-right image of Figure (6.1). Fortunately the eye placement is often still correct as the model is able to deform sufficiently to accommodate the mistake. Further evidence of difficulties acquiring the pitch can be seen in Figure (6.2), where the tracking results for videos isolating the roll, yaw and pitch are

shown over time for four participants. The graphs do show however that while tracking large rotations can produce inaccurate results, the 2.5D CLM is able to recover quickly once the rotation values return to less extreme orientations.

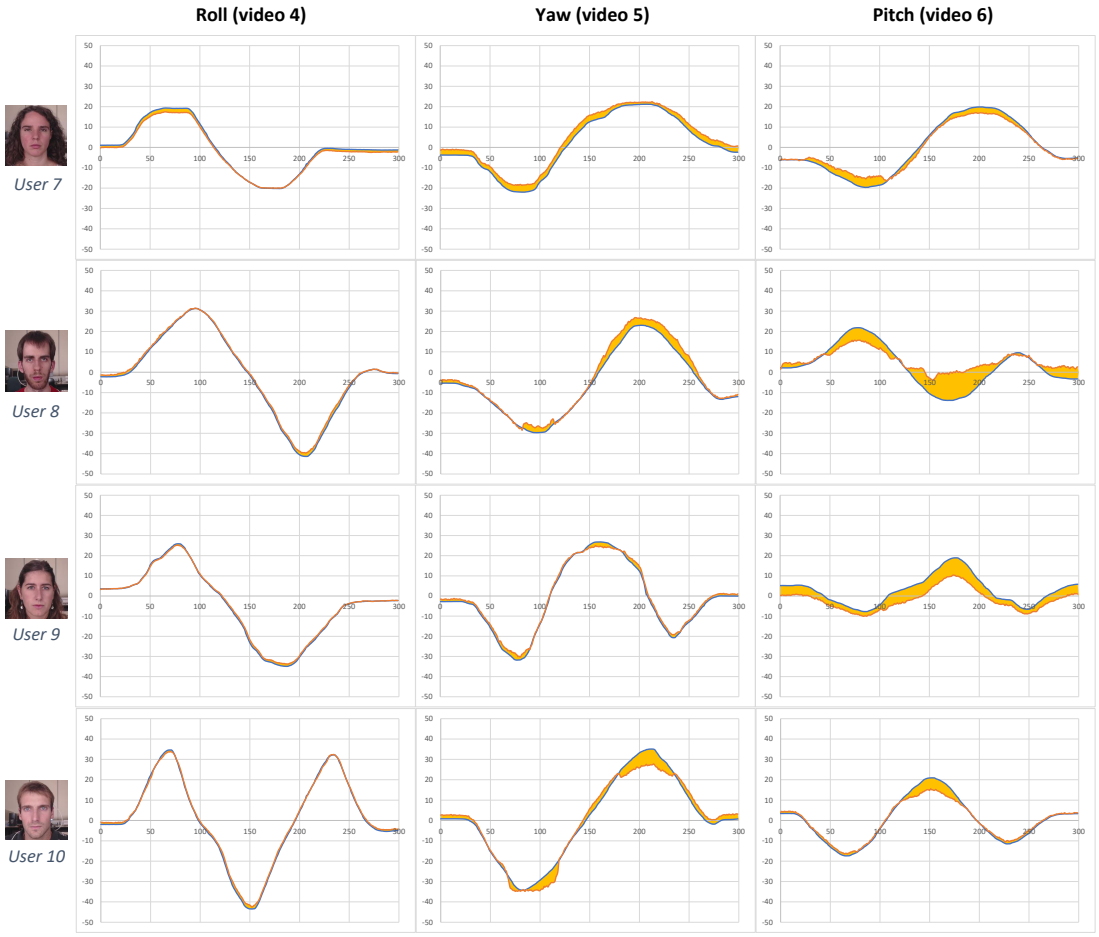


Fig. 6.2 Sample errors (in degrees) of roll, yaw and pitch over time on the UPNA dataset. The blue and orange lines show the ground truth and estimated values respectively, with the yellow area representing the error on each of the 300 frames of video. The participants were directed to perform isolated movements of roll, yaw and pitch on separate videos and these are compared here for four users. The largest errors occur for all 3 rotation axes when furthest from the front-facing position (0°), although it is clear that the tracking is able to recover well after these large rotations. The pitch is particularly difficult to determine when the participant has facial hair obscuring the outline of the face such as *User 8*.

Since it is not standard to report translation errors within the literature, there is no comparison metric available. However, since an inaccurate translation estimation could potentially have a negative effect on the gaze estimation it is

	Head Rotation Error (°)			
Tracker	Median Roll	Median Yaw	Median Pitch	Median (All)
2.5D CLM (MOSSE)	0.78	**1.65**	**2.04**	1.33
2.5D CLM (LNF)	**0.75**	1.73	**2.04**	**1.20**

Table 6.2 Median rotation errors on the UPNA dataset

	Head Translation Error (mm)					
	x		y		z	
Model	mean	median	mean	median	mean	median
2.5D CLM (MOSSE)	2.22 ± 5.31	1.08	2.03 ± 3.77	1.06	25.91 ± 27.74	18.77
2.5D CLM (CCNF)	**1.50 ± 1.61**	**1.06**	**1.26 ± 1.24**	**0.95**	**20.49 ± 14.88**	**15.70**

Table 6.3 Translation errors on the UPNA dataset

interesting to know how successful it is. The translation accuracy is reported in Table (6.3).

While the estimates for both x and y translations are very close to the ground truth, the z-translation has significantly larger errors. This is perhaps to be expected as monocular systems, even in humans who have lost the sight in one eye, struggle with depth perception. This is made significantly more complex by the fact that the model is allowed to deform and thus a small change in depth can be misattributed as a facial deformation. Low median errors for the translation parameters demonstrate that the tracker is robust. However one potential cause for concern is how much the z-translation, if frequently up to $2cm$ displaced from the recorded ground-truth, might have a negative effect on the eye-gaze estimate which relies so heavily on the head-pose.

The datasets obtained via low cost RGB-D cameras are not as precise as the

'flock-of-birds' tracker and have quite a large error range. Therefore as rotation values are not heavily dependent on absolute values only they are considered for these datasets. All frames of the EYEDIAP dataset were used where the pose remained within rotation values of $-30°$ and $+30°$ and where ground truth data was available. Any rotation outside of this set is extremely unlikely to occur in a mobile setting and only provides enough data for one eye with the other being mostly or totally occluded by the nose. In total 232,149 frames were available for the VGA video stream and 231,696 frames in HD. The ground truth estimation method used for the DMUHAG dataset provided sometimes undependable results and therefore only frames where the depth-to-model fit was below a threshold were used. This gave 23,041 frames available for comparison. Table (6.4) summarises the rotation errors using two kinds of filters with the 2.5D CLM.

While the MOSSE filters can provide similar tracking ability on close to front-facing users, they are simply not as robust as the LNF and are prone to failure more often resulting in wild fluctuations in the mean error. The DMUHAG dataset perhaps provides more realistic readings that would happen under regular gaze-tracking scenarios as the participants were simply asked to naturally follow the target and the screen size was significantly smaller. The EYEDIAP dataset features videos where the participants performed sometimes unnatural head movements as instructed. Still in terms of robustness the LNF filters are unmatched in their tracking ability.

6.2 Eye gaze estimation test results

The first test to assess to eye-gaze estimation abilities of the Constrained Geometric Binocular Model (CGBM) is on the Columbia dataset. Since the dataset is a collection of images where the participants place their head on a chin-rest, obtaining the head pose is occasionally prone to error due to parts of the face being occluded. A subset of 34 people from the test set (those without glasses) were evaluated. All angles of head pose ($-30°$ to $+30°$) were evaluated along with all 21 target points on a grid approximately $2.5m$ away from the participant giving 3570 gaze samples. The results can be seen in Table (6.5) for both the default estimate of the optical axis (OA) and an a visual axis (VA) example with $5°$ turn inward (θ) toward the nose for each eye and $2°$ tilt downward (ϕ). Of course these

	Head Pose Error (°)							
	Roll		Yaw		Pitch		All	
Model	M	Mdn	M	Mdn	M	Mdn	M	Mdn
								EYEDIAP (VGA)
2.5D CLM (MOSSE)	9.38	1.33	9.55	1.86	17.78	10.06	12.24	2.55
2.5D CLM (LNF)	**2.16**	**0.95**	**2.71**	**1.74**	**6.22**	**4.99**	**3.70**	**1.86**
								EYEDIAP (HD)
2.5D CLM (MOSSE)	11.38	3.04	11.54	3.51	20.98	12.37	14.63	5.81
2.5D CLM (LNF)	**2.67**	**1.42**	**3.67**	**1.75**	**8.70**	**6.64**	**5.01**	**2.29**
								DMUHAG
2.5D CLM (MOSSE)	**2.03**	**1.51**	3.90	3.21	8.05	7.19	4.66	3.04
2.5D CLM (LNF)	2.10	1.62	**2.73**	**2.35**	**6.85**	**5.97**	**3.90**	**2.82**

Table 6.4 Mean (M) and Median (Mdn) rotation errors on the EYEDIAP and DMUHAG datasets

	Angular Gaze Error (°)	
Model	Mean	Median
3DMM	**8.87**	**7.54**
kNN	19.9	19.5
CGBM (OA) (***this work***)	13.15	12.08
CGBM (VA) (***this work***)	12.25	10.27

Table 6.5 Angular Gaze Errors on the Columbia dataset. 3DMM (Wood et al., 2016*a*), kNN (Wood et al., 2016*b*)

are just estimates for the average person and if the real values could be obtained for all participants this would likely increase the success of the model.

For comparison the results reported in Wood et al. (2016*a*) are also displayed which were tested on the same 24 people although only 680 images were used (20 per person). It is unknown what criteria was used to select the images. The first model for comparison is a 3D Morphable Model (3DMM) of the eye (Wood et al., 2016*a*). The model takes several seconds to evaluate the gaze so is not currently suitable for a mobile device but is included so as to compare with the current state-of-the-art. The second model is an appearance-based method which utilises a k-Nearest Neighbour (kNN) approach (Wood et al., 2016*b*).

The median value is again included as incorrectly determined head-pose and eye-gaze value estimations can cause large changes in the mean. Since the dataset features very large images, the 3DMM can utilise the extra detail to gradually refine the fit until convergence. While the CGBM is no match for the 3DMM in terms of accuracy it still measures up well and is of course over 100 times faster and so can be used in real-time.

It is interesting to observe the vertical and horizontal gaze errors individually. As can be seen in Table (6.6), the vertical and horizontal errors are quite similar. The horizontal component can be subject to large errors but still maintains a lower median value than the vertical dimension. Since the range is much smaller

	Angular Gaze Error (°)			
	Horizontal		Vertical	
Model	Mean	Median	Mean	Median
CGBM (OA)	**8.11 ± 7.83**	5.72	8.64 ± 6.18	7.70
CGBM (VA)	8.11 ± 8.41	**5.41**	**7.58 ± 5.54**	**6.44**

Table 6.6 Horizontal and vertical gaze errors on the Columbia dataset

in the vertical dimension this implies that the tracker can less easily discriminate in cases when the user is looking up or down. There are a few potential reasons for this – the eyelids often occlude the top and bottom of the iris and though they are part of the model they can vary considerably from person to person. Secondly, the head pose results have shown that the pitch of the head is the most difficult rotation angle to determine, and this of course will have a bearing on the vertical component of the gaze vector.

Using the estimated visual axis offset the median value actually becomes lower in the horizontal dimension even though the standard deviation increases. This implies that the visual offset is vitally important for each individual with the majority of results improving and a minority of results becoming significantly worse.

Since the Columbia dataset features 5 different viewing angles, it is possible to observe how well the model compares when the face is viewed front-on and from varying degrees to the side. Table (6.7) shows that the errors are largest when the head is viewed from an angle furthest from front-facing and some examples of the fitting can be seen in Figure (6.3). What is interesting is that the vertical error stays similar throughout, while the horizontal error deviates from excellent to very poor. Since the target placements are identical between the views, this implies that potentially the rotation of the head is inhibiting the trackers ability to pick up the required eye features. Texture patches of the eye are created from one viewing angle, but perhaps the change in texture between head poses would benefit from a multi-view model as seen with the head pose tracker. Another possibility is that

Fig. 6.3 Examples of model fitting on the Columbia dataset. When the user is not front facing the eye-gaze estimation task becomes more difficult. The top row shows fits with low error and bottom row with high angular error, which may be the result of an inaccurate starting head pose.

when the eye is rotated to its extremes the iris becomes significantly occluded and only one side becomes visible. The largest eye rotation in the test is potentially 45° in the horizontal and the synthetic eye used to generate the gaze manifolds only had a range of ±32° which likely accounts for some of the error. Fortunately such viewing angles are not commonplace on any typical eye-gaze task let alone on a mobile device. The synthetic model may also have other limitations such as an inability to adapt to some eye shapes.

The EYEDIAP dataset provides two different video streams which challenge the models in different ways. First, a very low resolution video captured of the head and eye and secondly, higher definition videos but with all images captured from the side (approximately 15° to 40°). Since the CGBM requires both eyes to be visible only portions of the available video data where this is the case are used. Still, for the VGA sequence of videos, 207,804 samples were available. A number of comparison metrics are provided from the literature (Wood et al., 2016*a*) and are shown in Table (6.8).

Both versions of the CGBM outperform the state-of-the-art models currently available for person independent gaze estimation. It is likely the combined binocular restriction along with the mainly close-to front-facing participants have provided ideal conditions for the new model. Additionally, the 3DMM does not

	Viewing Angle (Yaw)				
Model	−30°	−15°	0°	15°	30°
					Horizontal
CGBM (OA) (*this work*)	11.89	8.39	**4.44**	6.61	9.23
CGBM (VA) (*this work*)	12.10	8.36	**3.89**	6.80	9.41
					Vertical
CGBM (OA) (*this work*)	8.53	9.13	9.33	8.72	**7.47**
CGBM (VA) (*this work*)	**7.00**	7.48	7.50	8.25	7.65
					Combined
CGBM (OA) (*this work*)	16.01	13.63	**11.06**	12.03	13.01
CGBM (VA) (*this work*)	14.92	12.28	**9.11**	11.74	13.18

Table 6.7 Gaze errors on the Columbia dataset for differing viewing yaw angles. Mean Angular errors displayed

perform as well even though it takes several seconds to compute a result. Since the video quality is quite low it is likely that no additional benefit can be gained from the denser model. The kNN (Wood et al., 2016*b*), CNN (Zhang et al., 2015), RF(Sugano et al., 2014), ALR (Lu et al., 2014) methods are all appearance-based and tend to perform best when trained under the same lighting conditions as the test set. Additionally in these results the appearance-based models utilised the known 3D head pose to normalise the image and evaluate the results whereas the CGBM inherently determines its own pose alongside the head-pose model. A suitable comparison metric has been added to the bottom rows of Table (6.8) where each frame is initialised with the pose estimate supplied with the dataset (derived from depth data). Although this known pose is itself subject to noise, the table shows that an improved head pose estimate can have a positive effect

	Angular Gaze Error (°)	
Model	Mean	Median
3DMM	9.44	8.63
kNN	21.49	20.93
CNN	10.5	-
RF	12.0	-
ALR	12.7	-
CGBM (OA) (*this work*)	9.22	7.99
CGBM (VA) (*this work*)	**8.07**	**6.99**
CGBM (OA) with depth stream pose (*this work*)	9.21	8.00
CGBM (VA) with depth stream pose (*this work*)	7.97	**6.88**

Table 6.8 Gaze errors on the EYEDIAP (VGA) dataset. Results obtained from Wood et al. (2016*b*) and Zhang et al. (2015), with median errors presented where available. 3DMM (Wood et al., 2016*a*), kNN (Wood et al., 2016*b*), CNN (Zhang et al., 2015), RF (Sugano et al., 2014), ALR (Lu et al., 2014). The bottom two rows used the supplied depth stream pose from the dataset as a starting point on each frame.

on the CGBM result. A shape-based model used for tablets were also set up for testing (Wood and Bulling, 2014), however the software provided by the authors relies heavily on Haar detectors to find the eyes which failed regularly under head pose changes and when detection did occur it failed to obtain any meaningful gaze estimation (typically $> 30°$ error). This is likely due to the requirement of needing a very high resolution for the eye regions and a stable head pose. Additionally it has only been shown to work when the eyes are approximately

20*cm* from the screen which is unviable. Similar results were seen in (Zhang et al., 2015) with mean errors of around 47.1° on the MPIIGaze image dataset, although it is possible that with some careful engineering to the software the results could be improved. Nevertheless, the success of both the 3DMM and CGBM have shown that model and shape-based approaches can match and even exceed the results of the appearance-based methods in a person-independent and cross-dataset context.

Fig. 6.4 An example of model fitting on synchronous frames of the EYEDIAP dataset. The left hand side shows the VGA feed and the right hand side shows a (cropped for presentation) image from the HD stream. The red and blue lines represent the ground truth and estimated gaze respectively.

For the EYEDIAP HD videos 201,104 frames of video were available for eye gaze estimation, based on the same criteria as the VGA set. The results for the EYEDIAP HD videos are not quite as effective however they still match the 3DMM in a fraction of the time as seen in Table (6.9). This is similar to the Columbia dataset results where off-angle faces perform more poorly in comparison to front-faces. Since the HD camera was set to the side of the participant the dataset in entirely comprised of angles approximately 15° to 40°. Although it is unlikely for these head poses to be commonplace when using a tablet device (since the camera is often in-line with the screen) it is still a hindrance when utilising the CGBM 'in-the-wild'. An example of the head-pose and eye-gaze fit on synchronous frames can be seen in Figure (6.4).

The final experiment specifically for eye-gaze is to analyse CGBMs effectiveness under different use-cases of a mobile device. The DMUHAG dataset is used which has an equal number of targets for each user under a *'static'* context where

	Angular Gaze Error (°)	
Model	Mean	Median
3DMM	**11.00**	10.4
kNN	29.39	28.62
CGBM (OA) (*this work*)	13.59	12.39
CGBM (VA) (*this work*)	11.17	**9.74**
CGBM (OA) with depth stream pose (*this work*)	13.13	12.10
CGBM (VA) with depth stream pose (*this work*)	**10.60**	**9.44**

Table 6.9 Gaze errors on the EYEDIAP (HD) dataset. 3DMM (Wood et al., 2016*a*), *k*NN (Wood et al., 2016*b*). The bottom two rows used the supplied depth stream pose from the dataset as a starting point on each frame.

	Angular Gaze Error (°)					
	Horizontal (°)		**Vertical** (°)		**All** (°)	
Model	Mean	Median	Mean	Median	Mean	Median
						DMUHAG (*static*)
CGBM (OA) (*this work*)	3.96	2.83	**5.39**	**4.52**	**7.39**	**6.73**
CGBM (VA) (*this work*)	**3.82**	**2.66**	6.76	6.45	8.46	7.83
						DMUHAG (*dynamic*)
CGBM (OA) (*this work*)	**4.92**	3.65	**6.36**	**4.92**	**8.87**	**8.00**
CGBM (VA) (*this work*)	5.10	**3.61**	7.71	7.21	10.21	9.72

Table 6.10 Gaze errors on the DMUHAG dataset under *static* and *dynamic* use

the tablet was placed on a table and the *'dynamic'* one where the tablet is held in the hands. The results can be seen in Table (6.10).

As can be seen from the results, the CGBM performs well in the 'mobile' case, especially in the horizontal component of gaze. Under *static* conditions the models do perform better, but only marginally so. The vertical values are the cause of the majority of the inaccuracy here and it is almost certainly a direct consequence of a failure to correctly determine the head pitch as previously seen in Table (6.4). Unfortunately this is currently a severe limitation of the model as it is the pitch of the head that can have the most significant changes as the user adjusts their body posture for comfort.

As an aside the results show that there was no clear difference between those wearing glasses (mean 9.31, median 8.91) to those without (mean 9.42, median 8.38) which is surprising, however the number of samples was very small. Still it is a promising result and since the model can incorporate a specific angular offset it may be possible to tune the visual axis specifically for a user with glasses. Future investigations are required to show if this is statistically valid amongst the

	Frames Per Second (*fps*)					
	EYEDIAP (VGA)		**EYEDIAP (HD)**		**DMUHAG**	
Model	HP	HP & EG	HP	HP & EG	HP	HP & EG
MOSSE	**42.45**	**39.79**	**34.28**	**29.71**	**33.09**	**23.63**
LNF	31.84	28.75	27.86	21.96	26.12	20.15

Table 6.11 Average frames-per-second *fps* on various datasets. Labels correspond to Head Pose only (HP) and Head Pose with Eye Gaze (HP & EG)

wider population.

It is of interest to know how efficiently the algorithm performs on a tablet device using the internal battery only. The code-base uses the OpenCV library alongside a collection of self-developed libraries in C++. No significant optimisation steps were taken during development and there are likely to be many opportunities for parallelisation that were not taken, for example when obtaining the local texture responses for each point. The frames-per-second (*fps*) computation time was calculated using high performance timers on a *Microsoft Surface Pro 3* with an Intel i5-4300U CPU and 8Gb RAM that was released in 2014. The timer ran over the length of several videos of the EYEDIAP and DMUHAG datasets with the results presented in Table (6.11). The disparity between datasets comes mostly from the loading of the larger images and the cropping and rescaling of the various target features. That is to say, when the head is larger within the image frame, computation time increases. The MOSSE filters processed the frames quicker than the LNF model as expected but not significantly so. While they were the only filter to consistently achieve over 30*fps* on the head-pose tests, they were not as robust as the LNF filters and therefore the speed gain probably does not justify their use. The complete model in each case performing head-pose and eye-gaze hit the 20*fps* mark which is suitable for real-time purposes. The tablet reported during the test that it was utilising under 40% of the available system resources. As tablets are becoming more powerful each year, this shows that the proposed model is perfectly suited to a mobile device and sufficiently meets the minimum requirements for real-time use.

6.3 General discussion

Overall the newly proposed trackers for both the head-pose and eye-gaze method built on top of it have shown to perform well and in some cases outperform the current state-of-the-art seen in the literature. The experiments have determined that the proposed models have met the desired aims **A1** and **A2** outlined in section (1.3) – firstly, that the combination of the 2.5D CLM for head-pose estimation and the CGBM for gaze evaluation allow for the user to undergo free-head movement and secondly, that the model can indeed achieve real-time performance under testing on a mobile device. The 2.5D head pose estimator has a significant advantage over other models in that it directly optimises the rotation and translation parameters, rather than being a two-stage process like many other models. The weakest rotation angle was consistently the pitch of the head, perhaps due to lack of relative depth around the nose region. Although a person's nose shape remains relatively static the 3D model is capable of a large amount of deformation around the nose to accommodate many different people. The z-translation component suffers from a similar problem where the optimisation process can stretch and squash the face as it needs, rather than move correctly along the z-dimension. Both of these issues could potentially be alleviated by learning about the users face shape over time so that the inter-person deformation is removed and only the deformation specific to that individual is kept.

From the three head-pose datasets it was clear that of the two varieties of filter tested, the LNF texture model is easily the most robust. Both filter types produced accurate results under small yaw rotations but under larger rotation changes the MOSSE filters were prone to tracking failure with the LNF filters performing considerably better. The MOSSE filters do have the advantage of running much quicker and using less computation resources. This proposes that under simpler conditions where the user is close to front-facing the MOSSE filters could be used satisfactorily with the more complex filter being used only under more challenging conditions. Since many mobile users are likely to be front-facing for the majority of the time due to the close relation between the camera and the display, this seems a viable compromise.

There have however been several key areas that have been determined where the gaze-tracker does not perform as well. Generally speaking, the vertical com-

ponent of the gaze-vector is more difficult to reliably track and further investigation is required regarding how this can be improved. The horizontal component of the gaze-vector under the model can be deemed a success, however under large changes in yaw this is the value that accumulates the largest errors. Since the head-pose and eye-gaze have been shown to be intrinsically linked it is not a stretch to consider that further improving the head pose estimation under larger rotations may also improve the eye-gaze estimation. This could perhaps be achieved by tracking additional *fixed* points on the side of the face such as the ears rather than the cheek outline, although there are no guarantees that these would always be visible.

Three key factors were outlined in Chapter 3 – the *Camera, Environment* and *User*. Regarding the quality of the *Camera*; or more specifically the resolution of the head and eye within the images and video; there appeared to be no conclusive evidence that this had any effect on either the head-pose or the eye-gaze estimation. In fact some of the the more impressive eye-gaze results came on the VGA feed of the EYEDIAP dataset, which had the lowest resolution of all the tested data. For the head-pose estimation the face image is scaled down to a relatively small standard size for efficiency and this could be the reason that there is no discernible difference between the different cameras. It is likely however that the lower quality starting image makes it harder for the texture patches to isolate their target with sub-pixel accuracy as seen with the 3DMM and of course this can affect the robustness of the tracker making it more prone to failure. Low quality images though can come from many other issues such as low lighting or a small image sensor on the camera itself.

Under the *Environment* category the CGBM performed marginally worse when held in the hands compared to when staticly placed on a desk within the DMUHAG dataset showing that the algorithm is easily transferable to any mobile device provided the relations between the camera and screen are fixed. This is the case in all modern tablets, laptop and phones as the cameras are built into the device with the vast majority having a fixed focal length. Admittedly, the sample size of the DMUHAG dataset is relatively small and videos were recorded in the same location in both sets of data. Uncontrolled environments are prone to changes in lighting and cast shadows on the face. Though DMUHAG was not recorded under a controlled setup, it will be interesting to see how the model

works under more challenging scenarios.

In terms of the *User* variation many datasets were used to cover as many different people as possible. Although difficult to quantitatively assess, the head pose tracking was empirically observed to perform worse for people with facial hair. This had the effect of incorrectly evaluating the entire outline of the face causing significant errors, particularly in pitch angle. Thankfully the location of the eyes within the image was still obtained correctly most of the time allowing for eye-gaze estimation to be conducted, although this was prone to error. For eye-gaze, the new model performed approximately the same for people wearing glasses although this cannot be considered statistically significant on such a small sample.

6.4 Summary

In this chapter a number of different datasets were used to test both the head-pose and eye-gaze estimation abilities of the newly proposed 2.5D CLM and CGBM. The proposed models performed well against other models seen in the literature and it has been shown that the model can feasibly be used within a mobile context. A number of weaknesses were also discussed in depth and reasons have been suggested regarding the possible reasons why these may have come about.

The next chapter will summarise the contributions made within the thesis as a whole and will suggest a number of interesting avenues for future work which have been determined from the results presented here.

Part V

Conclusions & Future Research Work

Chapter 7

Conclusions

The aim of this work was to develop a model for eye-gaze estimation on commodity mobile devices. Early on it was determined that since the head needed to be able to move freely; particularly when the device was being held in the user's hands; a novel method to estimate the head-pose *for* eye-gaze was required. The two models are perfectly synchronised in order to share a common origin and set of pose parameters. This chapter summarises the work undertaken and suggests potential areas for future research.

7.1 Contributions

Working towards the aims **A1-A2** and objectives **O1-O4** outlined at the beginning of this thesis (in section 1.3) led to many notable contributions that shall be discussed here. The first part of this thesis focussed on providing background information for the task of estimating the eye-gaze direction and point-of-regard from a single monocular camera using only visible light. Understanding the underlying structure and shape of the eye was key in developing a binocular approach to the problem, requiring both eyes to be looking at the same point in 3D space. Two main varieties of eye-gaze estimation – interpolation-based and geometric methods were reviewed in detail to complete objective **O1**. The paradigm that made the most convincing case was the geometric one for several reasons. Firstly, under mobile conditions the relationship between the screen and the camera is

fixed and determined through controlled calibration methods, which simplifies many of the complexities that typically arise with geometric methods. Secondly, since the head-pose is also to be determined importance was placed on ensuring the two models could be easily linked. Head pose estimation, at its core, is a geometric task and allowing the entire proposed solution to share a set of common parameters became one of the defining features of this work.

The second part of this thesis detailed a number of tools and datasets that could be used in the completion of the work. Since no suitable dataset was available for the analysis of head-pose and eye-gaze models on a mobile device, a small dataset was created called DMUHAG, completing objective **O3**. Eleven participants took part in the recordings with their head shape being scanned to obtain their 'real-3D' face shape. Furthermore, two videos were recorded for each user on an RGB-D camera where the participants were asked to look at a series of targets on the mobile device in two modes – first with the tablet *staticly* placed on a desk and secondly, in a *dynamic* situation where the tablet was held in the hands. Additionally, the criteria were set out for how the testing was to take place. In particular there was to be a focus on the *Camera*, *Environment* and *User*, as these are the variables that, tested vigorously, would show where the successes and failures of the proposed models lie.

Part III of the thesis focussed on objective **O2** and presented the proposed solutions to the problems of head-pose and eye-gaze estimation on mobile devices in detail. *Chapter 3* introduced the 2.5D Constrained Local Model (2.5D CLM) – its main contributions were:

- *A model which combined a 3D shape point model with 2D texture information that provides direct estimation of the pose parameters, avoiding the need for additional optimisation strategies.*
- *The potential use of a number of texture filters including a novel optimisation of a MOSSE filter trained from a number of different viewpoints.*
- *A heuristic that globally aligns the face shape consistently. It dynamically updates to improve the robustness of the tracking within fast-changing images and provides a method for detecting tracking failure.*
- *The analytical derivation of a Jacobian matrix, which describes how changes*

in the parameters of the model create changes in the point positions within the image through a full-perspective camera model

- *The creation of the 3D shape model through careful alignment of the training shapes via the inner eye-corners. This reduced the complexity of the model and allowed for simple synchronisation with the position of the eyes.*

Chapter 4 introduced the Constrained Geometric Binocular Model (CGBM) – its main contributions being:

- *The implementation of a geometric model for eye-gaze that combines a 3D shape point model with 2D texture information. Like the head-pose model, it also provides direct estimation of the pose parameters and the screen coordinates simultaneously.*
- *The creation of a binocular shape model that symbiotically links the head-pose and eye-gaze models through a shared pose and local origin.*
- *The development of a multi-dimensional and multivariate eye-orientation manifold.*
- *The derivation of a Jacobian matrix which describes how changes in the screen coordinates create changes in the point positions within the image through the use of a full-perspective camera model.*

The final critical part of the thesis was to evaluate the head-pose and eye-gaze models for objective **O4**. It was shown that the head-pose and eye-gaze models are able to run in real-time on a commodity tablet device, with the joint algorithm running at just over 20*fps* using the most complex texture filters under the most challenging conditions. The gaze framework therefore meets the real-time requirement **A2** set out at the beginning of this thesis. The head-pose estimation model; the 2.5D CLM; was successfully able to outperform 2D ASM and AAM trackers that are supplemented with the POSIT algorithm on the UPNA dataset. Similar results were measured on RGB-D datasets, however while the LNF texture filters performed robustly, the efficient MOSSE filters struggled when the head rotation was extreme, resulting in a loss of tracking and a poorer mean angular error. When tracking was not lost, only minor differences were observed

between the two filter outputs, particularly on close-to front-facing images. Both filters struggled on occasion to deal with changes in the pitch angle estimate. Since the MOSSE filters were shown to run faster and as such use less computational resources per image frame, there is a likely to be a use-case for both filters depending on the complexity of the incoming image and the requirement for efficiency that is vitally important on mobile devices.

The proposed eye-gaze model; the CGBM; successfully matches or outperforms a number of appearance-based methods on the EYEDIAP and Columbia datasets. While simpler shape and model-based approaches were unable to generalise to a number of different settings, the new model utilises a large sample set of training data to build a user and pose-independent manifold. The manifold significantly constrains the problem through the requirement of a known head-pose and the need for both eyes to observe the same location on a known monitor plane. When held in the hands the gaze estimation errors were only marginally worse than in the static case on the DMUHAG dataset. This meets the set aim **A1** of creating a gaze estimation model that allows for free head movement. However amongst the other test conditions, under extreme rotations the combined head-pose and eye-gaze method experienced large errors.

The following section provides a critical review of the work carried out based on the original research question, and suggests possible avenues for future work in the subject.

7.2 Critical review & future work

The key question asked at the start of this thesis was

> *Can a reliable stream of gaze data be generated from a mobile device?*

To help answer this question several underlying questions were posed. Now that the new models have been evaluated it is possible to revisit these questions to provide critical analysis.

> *Which current algorithms can be adapted and embedded to run in real-time on an unmodified mobile device?*

Two eye-gaze paradigms were assessed – the interpolation approach and the geometric approach. From a review of the literature it was deemed that either method could make for a successful eye-gaze estimation strategy on an unmodified mobile device provided that the head-pose was known. Geometric methods inherently require the head-pose whereas the most recent appearance-based methods have shown great promise because they have utilised a known head-pose to rectify the image before estimating the gaze vector. Both methods can be built to be pose and person invariant under passive illumination, although the most common methods seen in the literature were appearance-based. The difficulty is normally in acquiring the large number of training samples required, although this has been mitigated through the use of synthetic eye models which produce lifelike representations and can be manipulated to provide millions of samples as required. Other methods that show promise are the geometric-based methods that rely on shape and feature models of the eye. The simplest models use potentially naive assumptions, for example, the iris outline can be extracted from the image to approximate the iris normal, which is deemed to represent the gaze direction. As the features and mechanisms of the eye are well known it was determined that the geometric approach for eye-gaze was the best suited for the task at hand as it has the potential to be tightly coupled with a geometric head pose estimate. This does not preclude the use of the same synthetic eye datasets that the appearance-based methods benefit from, where instead they can be exploited as training data for feature detection, helping to alleviate some of the difficulty in estimating the iris and eyelid features under passive light conditions.

> *Assuming a stream of gaze data can be generated, what are the limiting factors and how much do they influence?*

The three primary factors that influence the gaze estimation were considered to be related to the *Camera, Environment* and *User*. For the camera, one of the main issues in determining an accurate estimate of the gaze is in the ability to extract features of the iris with the limited pixel density around the eye region. This happens when low resolution cameras are used, or the head is too far away from the camera. In other cases even higher resolutions may not resolve the issue since the field-of-view of front-facing cameras are sometimes wide-angled on mobile devices with the head and eye region only taking up small real-estate

from the incoming camera image. As the pixel region of the eye can be very small, features detected; even a few pixels offset from the true value; can result in a large gaze error. Unfortunately the iris can be one of the most difficult features to observe in its totality, as in the majority of cases the iris edge is partially occluded by the eyelids. That being said, it has been shown that even on relatively low quality images (within the EYEDIAP dataset) the gaze can be estimated with a mean error of around 10° when subject to large head pose changes. When comparing the environment setup with the tablet device held in the hands, there was only a small drop in the mean angular error showing that the gaze stream is robust to the increased pose changes from the user relative to the camera. From this we can determine that the head-pose tracker works robustly in cases where the head remains mostly front-facing. The largest errors were seen in the dataset when a loss of head-pose tracking occurred and this happened most frequently on the EYEDIAP dataset where the head motions were often extreme. It will be interesting in the future to see if the failure to fit the gaze correctly can be determined robustly, since if the eyes can not be synchronised within a reasonable tolerance this may be the first major give-away that the head-pose has failed. Similar head poses were not observed on the DMUHAG dataset other than a consistent downward pitch that sometimes the head-pose tracker failed to correctly estimate.

Tracking of the head-pose was empirically observed to be worse with users who had facial hair obscuring the true outline of the face. Other examples, including cases where people were wearing glasses, produced results that were not significantly different than the group with no glasses, although the amount of data available for comparison was limited. Another potential reason for failed eye-gaze estimates is that the specific eye-shape of the user does not match well to the generic manifold that has been built for all users. Although we are restricted to eye regions that can be generated by the synthetic eye-model, it may be possible to identify several common eye types with the best fit being chosen for each particular user. A better alternative may be to attempt to learn a specific shape model for each user through gradual manipulation of the eye-gaze manifolds; perhaps through machine-learning techniques; although care has to be taken not to introduce laborious calibration requirements.

Can we track the head (and in turn the eyes) quickly and accurately

enough with limited computational power to collect gaze data?

The head-pose and eye-gaze estimation joint-process was successfully able to run on a commodity tablet in all cases at over 20 *fps* ensuring that the models are fast enough for real-time use. Lower quality video was processed faster than 40 *fps* using the efficient MOSSE filters, although the robustness of the solution suffered slightly on faces undergoing significant rotation from the frontal position. In addition, the results show that the mean angular errors of the head pose tracker match or in some cases beat the state-of-the-art. The work can be extended by using the many opportunities for parallelisation within the code to optimise the algorithm. Additionally, since the model does not require a specific texture model, numerous other methods for correlation can be explored along with varying sizes of the local search regions.

How does the change in the user's head pose affect our ability to estimate gaze?

When the head remained front-facing in the image the eye-gaze results remained at their most accurate. Fortunately, during the tests on the mobile device, when the tablet was held in the hands of the user the angular errors were only marginally worse. This fact seems independent of the translation values and rather it is under extreme rotation where the gaze tracking starts to fail, sometimes with very large errors. This may be because while translation may be frequent in this mode, rotation (particularly in yaw, which obscures part of the eye region) only occurs to a small degree. To this end we can say that the proposed model generally meets the targets it set out to achieve and performs well under a variety of differing head poses. However, it was demonstrated on the Columbia dataset that larger rotation in yaw frequently produced significant errors in the horizontal component of the gaze vector and this was speculated to be a result of the head pose tracker failing to sufficiently estimate the rotation. In that sense we can say that errors produced by the head pose tracker impinge the ability of the gaze-model to reliably estimate the point-of-regard. Future work will attempt to understand precisely under what conditions this occurs in an attempt to minimise the errors under pose variations.

There were only a limited number of occasions where the head (specifically the chin region) partially left the camera image, which first affects the head-pose tracking estimate. Since the eye-gaze relies upon the head-pose result this of course is likely to have an effect on the gaze-tracking estimate. Mobile devices are prone to this happening frequently since a number of body postures can be adopted by the user and perhaps alternative points can be utilised such as those around the ear, although these too are often not visible within the image. It may be possible to densely position the points around only the eyes and nose, although, as has already been suggested, the shorter the distance between the points the more difficult it is to estimate angles such as the pitch. An alternative is to utilise the explosion of commodity RGB-D cameras that are being built-in to modern laptops, with the additional depth data potentially allowing for a more robust head-pose estimator to be built. Additionally, since these devices work through the use of infra-red it is likely that the pupil (and iris) features will be able to be estimated more accurately. There is no reason to suggest the Constrained Geometric Binocular Model would not produce even more impressive results under this (more constrained) scenario.

Which parameters of the framework need to be calibrated for individuals and how often does this need to take place?

No specific calibration is required as the model is fully pose and person independent. That is not to say that there is no potential for the method to be specifically calibrated for a user in a one-time calibration provided sufficient training data was available. The thesis described how either the optical or visual axis could be utilised and it was demonstrated that with a fixed common offset set for all users the overall mean angular error could be reduced. It was noted however that while some users benefited from the angle offset, others were significantly and consistently worse off. Once the offset for a person is known there would be no need to perform the calibration again and therefore perhaps the user would tolerate a more demanding initial calibration procedure if the results were suitably impressive enough. On the first successful frame after the head has correctly been fit to a new user, the binocular model was dynamically built by shifting the distance between the inner eye-corners. This ensured a good match between the two models and required no additional input from the participant taking negligible computation time.

To summarise and focus the above answers toward the original key question, from the initial results we can say that the gaze stream generated frequently matches state-of-the-art methods under conditions with pose and person invariance *provided* the head rotation is not too excessive. While the accuracy does not match that seen in many commercial hardware, the evaluated real-time gaze can be used in a wide range of applications and scenarios typical on mobile devices such as modelling user attentiveness and interaction on websites and within video games. The results on a tablet-based dataset approximately matched those within non-mobile environments showing that the models are viable for real-time usage on a tablet device. In particular, the horizontal estimate of gaze generates good results and minimal errors whereas the vertical component is much more unreliable. This is potentially because the head pose estimate has the largest errors in pitch angle. Nevertheless, the benefits of the proposed model were observed with the geometric method suitably shifting the burden of calibration off of the user and onto a one-time calibration of the system setup. While a good estimate of the monitor position in relation to the camera can be difficult to acquire, many modern phones, laptops and cameras have their cameras in-built and planar to the screen itself, thus permanently fixing the calibration. This calibration need not be a burden on the end-consumer but instead could be meticulously estimated during factory creation, which would greatly enable eye-gaze on mobiles to become a new standard input-modality for end-users.

References

Ackland, S., Istance, H., Coupland, S. and Vickers, S. (2014), An investigation into determining head pose for gaze estimation on unmodified mobile devices, *in* 'Proceedings of the Symposium on Eye Tracking Research and Applications', ACM, pp. 203–206.

Ariz, M., Bengoechea, J. J., Villanueva, A. and Cabeza, R. (2016), 'A novel 2d/3d database with automatic face annotation for head tracking and pose estimation', *Computer Vision and Image Understanding* **148**, 201–210.

Baker, S. and Matthews, I. (2004), 'Lucas-kanade 20 years on: A unifying framework', *International journal of computer vision* **56**(3), 221–255.

Baltrušaitis, T., Robinson, P. and Morency, L.-P. (2012), 3d constrained local model for rigid and non-rigid facial tracking, *in* 'Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on', IEEE, pp. 2610–2617.

Baltrusaitis, T., Robinson, P. and Morency, L.-P. (2013), Constrained local neural fields for robust facial landmark detection in the wild, *in* 'Proceedings of the IEEE International Conference on Computer Vision Workshops', pp. 354–361.

Baltrušaitis, T., Robinson, P. and Morency, L.-P. (2016), Openface: an open source facial behavior analysis toolkit, *in* 'Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on', IEEE, pp. 1–10.

Baltru
vsaitis, T. (2014), Automatic facial expression analysis, PhD thesis, University of Cambridge.

Belhumeur, P. N., Jacobs, D. W., Kriegman, D. J. and Kumar, N. (2013), 'Localizing parts of faces using a consensus of exemplars', *IEEE transactions on pattern analysis and machine intelligence* **35**(12), 2930–2940.

Berrio, E., Tabernero, J. and Artal, P. (2010), 'Optical aberrations and alignment of the eye with age', *Journal of vision* **10**(14), 34–34.

Blanz, V. and Vetter, T. (1999), A morphable model for the synthesis of 3d faces, *in* 'Proceedings of the 26th annual conference on Computer graphics and interactive techniques', ACM Press/Addison-Wesley Publishing Co., pp. 187–194.

Böhme, M., Dorr, M., Graw, M., Martinetz, T. and Barth, E. (2008), A software framework for simulating eye trackers, *in* 'Proceedings of the 2008 symposium on Eye tracking research & applications', ACM, pp. 251–258.

Bolme, D. S., Beveridge, J. R., Draper, B. A. and Lui, Y. M. (2010), Visual object tracking using adaptive correlation filters, *in* 'Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on', IEEE, pp. 2544–2550.

Bradski, G. (2000), 'Opencv', *Dr. Dobb's Journal of Software Tools* .

Cerrolaza, J. J., Villanueva, A., Sukno, F. M., Butakoff, C., Frangi, A. F. and Cabeza, R. (2012), 'Full multiresolution active shape models', *Journal of Mathematical Imaging and Vision* **44**(3), 463–479.

Cerrolaza, J., Villanueva, A. and Cabeza, R. (2008), Taxonomic study of polynomial regressions applied to the calibration of video-oculographic systems, *in* 'Proceedings of the 2008 symposium on Eye tracking research & applications', ACM, pp. 259–266.

Chen, J. and Ji, Q. (2011), Probabilistic gaze estimation without active personal calibration, *in* 'Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on', IEEE, pp. 609–616.

Chen, Q., Matsumoto, K. and Wu, H. (2013), An iris outline tracker: for various eye shapes in a single video camera without infrared illumination, *in* 'Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on', IEEE, pp. 862–866.

Cheung, Y.-m. and Peng, Q. (2015), 'Eye gaze tracking with a web camera in a desktop environment', *Human-Machine Systems, IEEE Transactions on* **45**(4), 419–430.

Choi, S. and Kim, D. (2008), 'Robust head tracking using 3d ellipsoidal head model in particle filter', *Pattern Recognition* **41**(9), 2901–2915.

Cooley, J. W. and Tukey, J. W. (1965), 'An algorithm for the machine calculation of complex fourier series', *Mathematics of computation* **19**(90), 297–301.

Cootes, T. F., Edwards, G. J. and Taylor, C. J. (2001), 'Active appearance models', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **23**(6), 681–685.

Coutinho, F. and Morimoto, C. (2006), Free head motion eye gaze tracking using a single camera and multiple light sources, *in* 'Computer Graphics and Image Processing, 2006. SIBGRAPI'06. 19th Brazilian Symposium on', IEEE, pp. 171–178.

Cristinacce, D. and Cootes, T. (2006), Feature detection and tracking with constrained local models, *in* 'Proc. British Machine Vision Conference', Vol. 3, pp. 929–938.

Daugman, J. (2007), 'New methods in iris recognition', *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **37**(5), 1167–1175.

Davatzikos, C., Tao, X. and Shen, D. (2003), 'Hierarchical active shape models, using the wavelet transform', *IEEE transactions on medical imaging* **22**(3), 414–423.

Dementhon, D. F. and Davis, L. S. (1995), 'Model-based object pose in 25 lines of code', *International Journal of Computer Vision* **15**(1), 123–141.

Drewes, H., De Luca, A. and Schmidt, A. (2007), Eye-gaze interaction for mobile phones, *in* 'Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology', ACM, pp. 364–371.

Duchowski, A. (2007), *Eye tracking methodology: Theory and practice*, Vol. 373, Springer Science & Business Media.

Ebisawa, Y. (2009), Robust pupil detection by image difference with positional compensation, *in* 'Virtual Environments, Human-Computer Interfaces and Measurements Systems, 2009. VECIMS'09. IEEE International Conference on', IEEE, pp. 143–148.

Fanelli, G., Weise, T., Gall, J. and Van Gool, L. (2011), 'Real time head pose estimation from consumer depth cameras', *Pattern Recognition* pp. 101–110.

Fantz, R. L., Fagan, J. and Miranda, S. (1975), 'Early visual selectivity', *Infant perception: From sensation to cognition* **1**, 249–346.

Fischler, M. A. and Bolles, R. C. (1981), 'Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography', *Communications of the ACM* **24**(6), 381–395.

Funes-Mora, K. A. and Odobez, J.-M. (2015), 'Gaze estimation in the 3d space using rgb-d sensors', *International Journal of Computer Vision* pp. 1–23.

Goodall, C. (1991), 'Procrustes methods in the statistical analysis of shape', *Journal of the Royal Statistical Society. Series B (Methodological)* pp. 285–339.

Gross, R., Matthews, I., Cohn, J., Kanade, T. and Baker, S. (2010), 'Multi-pie', *Image and Vision Computing* **28**(5), 807–813.

Guestrin, E. and Eizenman, M. (2006), 'General theory of remote gaze estimation using the pupil center and corneal reflections', *Biomedical Engineering, IEEE Transactions on* **53**(6), 1124–1133.

Guestrin, E. and Eizenman, M. (2008), Remote point-of-gaze estimation requiring a single-point calibration for applications with infants, *in* 'Proceedings of the 2008 symposium on Eye tracking research & applications', ACM, pp. 267–274.

Hansard, M. and Horaud, R. (2010), 'Cyclorotation models for eyes and cameras', *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **40**(1), 151–161.

Hansen, D., Agustin, J. and Villanueva, A. (2010), Homography normalization for robust gaze estimation in uncalibrated setups, *in* 'Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications', ACM, pp. 13–20.

Hansen, D. and Ji, Q. (2010), 'In the eye of the beholder: A survey of models for eyes and gaze', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **32**(3), 478–500.

Hansen, D. and Pece, A. (2005), 'Eye tracking in the wild', *Computer Vision and Image Understanding* **98**(1), 155–181.

Hansen, D. W., Roholm, L. and Ferreiros, I. G. (2014), Robust glint detection through homography normalization, *in* 'Proceedings of the Symposium on Eye Tracking Research and Applications', ACM, pp. 91–94.

Henriques, J. F., Caseiro, R., Martins, P. and Batista, J. (2015), 'High-speed tracking with kernelized correlation filters', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **37**(3), 583–596.

Holland, C. and Komogortsev, O. (2012), Eye tracking on unmodified common tablets: Challenges and solutions, *in* 'Proceedings of ACM Eye Tracking Research & Applications Symposium, Santa Barbara, CA', ACM, pp. 1–4.

Huang, Q., Veeraraghavan, A. and Sabharwal, A. (2015), 'Tabletgaze: Unconstrained appearance-based gaze estimation in mobile tablets', *arXiv preprint arXiv:1508.01244* .

Huang, S., Wu, Y., Hung, W. and Tang, C. (2010), Point-of-regard measurement via iris contour with one eye from single image, *in* 'Multimedia (ISM), 2010 IEEE International Symposium on', IEEE, pp. 336–341.

Huey, E. B. (1908), *The psychology and pedagogy of reading*, The Macmillan Company.

Istance, H., Hyrskykari, A., Vickers, S. and Chaves, T. (2009), For your eyes only: Controlling 3d online games by eye-gaze, *in* 'IFIP Conference on Human-Computer Interaction', Springer, pp. 314–327.

Jianfeng, L. and Shigang, L. (2014), Eye-model-based gaze estimation by rgb-d camera, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops', pp. 592–596.

Karamchandani, H., Chau, T., Hobbs, D. and Mumford, L. (2015), Development of a low-cost, portable, tablet-based eye tracking system for children with impairments, *in* 'Proceedings of the international Convention on Rehabilitation Engineering & Assistive Technology', Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre, p. 5.

Kim, S.-T., Choi, K.-A., Shin, Y.-G. and Ko, S.-J. (2015), A novel iris center localization based on circle fitting using radially sampled features, *in* 'Consumer Electronics (ISCE), 2015 IEEE International Symposium on', IEEE, pp. 1–2.

Kirby, M. and Sirovich, L. (1990), 'Application of the karhunen-loeve procedure for the characterization of human faces', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **12**(1), 103–108.

Kohlbecher, S., Bardinst, S., Bartl, K., Schneider, E., Poitschke, T. and Ablassmeier, M. (2008), Calibration-free eye tracking by reconstruction of the pupil ellipse in 3d space, *in* 'Proceedings of the 2008 symposium on Eye tracking research & applications', ACM, pp. 135–138.

Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W. and Torralba, A. (2016), Eye tracking for everyone, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 2176–2184.

La Cascia, M., Sclaroff, S. and Athitsos, V. (2000), 'Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **22**(4), 322–336.

Langton, S. R., Honeyman, H. and Tessler, E. (2004), 'The influence of head contour and nose angle on the perception of eye-gaze direction', *Attention, Perception, & Psychophysics* **66**(5), 752–771.

Li, D., Winfield, D. and Parkhurst, D. J. (2005), Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches, *in* 'Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on', IEEE, pp. 79–79.

Lu, F. and Chen, X. (2015), 'Person-independent eye gaze prediction from eye images using patch-based features', *Neurocomputing* .

Lu, F., Okabe, T., Sugano, Y. and Sato, Y. (2011), A head pose-free approach for appearance-based gaze estimation., *in* 'BMVC', pp. 1–11.

Lu, F., Sugano, Y., Okabe, T. and Sato, Y. (2014), 'Adaptive linear regression for appearance-based gaze estimation', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **36**(10), 2033–2046.

Martins, P., Caseiro, R. and Batista, J. (2012), 'Generative face alignment through 2.5 d active appearance models', *Computer Vision and Image Understanding* .

Matthews, I. and Baker, S. (2004), 'Active appearance models revisited', *International Journal of Computer Vision* **60**(2), 135–164.

Merchant, J., Morrissette, R. and Porterfield, J. L. (1974), 'Remote measurement of eye direction allowing subject motion over one cubic foot of space', *Biomedical Engineering, IEEE Transactions on* (4), 309–317.

Miluzzo, E., Wang, T. and Campbell, A. (2010), Eyephone: activating mobile phones with your eyes, *in* 'Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds', ACM, pp. 15–20.

Model, D. and Eizenman, M. (2012), A probabilistic approach for the estimation of angle kappa in infants, *in* 'Proceedings of the Symposium on Eye Tracking Research and Applications', ACM, pp. 53–58.

Mora, K. A. F., Monay, F. and Odobez, J.-M. (2014), Eyediap: A database for the development and evaluation of gaze estimation algorithms from rgb and rgb-d cameras, *in* 'Proceedings of the Symposium on Eye Tracking Research and Applications', ACM, pp. 255–258.

Mora, K. A. F. and Odobez, J.-M. (2012), Gaze estimation from multimodal kinect data, *in* 'Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on', IEEE, pp. 25–30.

Morimoto, C. H., Amir, A. and Flickner, M. (2002), Detecting eye position and gaze from a single camera and 2 light sources, *in* 'Pattern Recognition, 2002. Proceedings. 16th International Conference on', Vol. 4, IEEE, pp. 314–317.

Nadaraya, E. A. (1964), 'On estimating regression', *Theory of Probability & Its Applications* **9**(1), 141–142.

Nagamatsu, T., Sugano, R., Iwamoto, Y., Kamahara, J. and Tanaka, N. (2010), User-calibration-free gaze tracking with estimation of the horizontal angles between the visual and the optical axes of both eyes, *in* 'Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications', ACM, pp. 251–254.

Nagamatsu, T., Yamamoto, M. and Sato, H. (2010), Mobigaze: development of a gaze interface for handheld mobile devices, *in* 'Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems', ACM, pp. 3349–3354.

Nakamatsu, Y., Takiguchi, T. and Ariki, Y. (2012), Gaze estimation using 3d active appearance models, *in* 'International Workshop on Nonlinear Circuits', Communications and Signal Processing.

National-Eye-Institute, T. (2013), 'Eye diagram. internet. retrieved march 2013'. **URL:** *"http://www.nei.nih.gov/photo/eyean/images/NEA09_72.jpg"*

Padeleris, P., Zabulis, X. and Argyros, A. A. (2012), Head pose estimation on depth data based on particle swarm optimization, *in* 'Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on', IEEE, pp. 42–49.

Papoutsaki, A., Daskalova, N., Sangkloy, P., Huang, J., Laskey, J. and Hays, J. (2016), Webgazer: scalable webcam eye tracking using user interactions, IJCAI.

Paquet, U. (2009), Convexity and bayesian constrained local models, *in* 'Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on', IEEE, pp. 1193–1199.

Park, K. (2007), 'A real-time gaze position estimation method based on a 3-d eye model', *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **37**(1), 199–212.

Paysan, P., Knothe, R., Amberg, B., Romdhani, S. and Vetter, T. (2009), A 3d face model for pose and illumination invariant face recognition, *in* 'Advanced video and signal based surveillance, 2009. AVSS'09. Sixth IEEE International Conference on', IEEE, pp. 296–301.

Pirri, F., Pizzoli, M. and Rudi, A. (2011), A general method for the point of regard estimation in 3d space, *in* 'Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on', IEEE, pp. 921–928.

Rusinkiewicz, S. and Levoy, M. (2001), Efficient variants of the icp algorithm, *in* '3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on', IEEE, pp. 145–152.

Saragih, J., Lucey, S. and Cohn, J. (2009), Face alignment through subspace constrained mean-shifts, *in* 'Computer Vision, 2009 IEEE 12th International Conference on', Ieee, pp. 1034–1041.

Saragih, J., Lucey, S. and Cohn, J. (2011), 'Deformable model fitting by regularized landmark mean-shift', *International Journal of Computer Vision* **91**(2), 200–215.

Sesma, L., Villanueva, A. and Cabeza, R. (2012), Evaluation of pupil center-eye corner vector for gaze estimation using a web cam, *in* 'Proceedings of the Symposium on Eye Tracking Research and Applications', ACM, pp. 217–220.

Sewell, W. and Komogortsev, O. (2010), Real-time eye gaze tracking with an unmodified commodity webcam employing a neural network, *in* 'Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems', ACM, pp. 3739–3744.

Shao, G., Che, M., Zhang, B., Cen, K. and Gao, W. (2010), A novel simple 2d model of eye gaze estimation, *in* 'Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2010 2nd International Conference on', Vol. 1, IEEE, pp. 300–304.

Shih, S. and Liu, J. (2004), 'A novel approach to 3-d gaze tracking using stereo cameras', *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **34**(1), 234–245.

Skodras, E., Kanas, V. G. and Fakotakis, N. (2015), 'On visual gaze tracking based on a single low cost camera', *Signal Processing: Image Communication* **36**, 29–42.

Smith, B. A., Yin, Q., Feiner, S. K. and Nayar, S. K. (2013), Gaze locking: passive eye contact detection for human-object interaction, *in* 'Proceedings of the 26th annual ACM symposium on User interface software and technology', ACM, pp. 271–280.

Snowden, R., Thompson, P. and Troscianko, T. (2012), *Basic Vision: an introduction to visual perception*, OUP Oxford.

Sugano, Y., Matsushita, Y. and Sato, Y. (2010), Calibration-free gaze sensing using saliency maps, *in* 'Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on', IEEE, pp. 2667–2674.

Sugano, Y., Matsushita, Y. and Sato, Y. (2014), Learning-by-synthesis for appearance-based 3d gaze estimation, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 1821–1828.

Takahashi, K., Nobuhara, S. and Matsuyama, T. (2012), A new mirror-based extrinsic camera calibration using an orthogonality constraint, *in* 'Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on', IEEE, pp. 1051–1058.

Tang, Y., Sun, Z. and Tan, T. (2011), 'Real-time head pose estimation using random regression forests', *Biometric Recognition* pp. 66–73.

Timm, F. and Barth, E. (2011), 'Accurate eye centre localisation by means of gradients.', *VISAPP* **11**, 125–130.

Torresani, L., Hertzmann, A. and Bregler, C. (2008), 'Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **30**(5), 878–892.

Torricelli, D., Conforto, S., Schmid, M. and D'Alessio, T. (2008), 'A neural-based remote eye gaze tracker under natural head motion', *Computer methods and programs in biomedicine* **92**(1), 66–78.

Valenti, R. and Gevers, T. (2011), 'Accurate eye center location through invariant isocentric patterns', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (99), 1–1.

Valenti, R., Sebe, N. and Gevers, T. (2012), 'Combining head pose and eye location information for gaze estimation', *Image Processing, IEEE Transactions on* (99), 1–1.

Villanueva, A. and Cabeza, R. (2007), 'Models for gaze tracking systems', *Journal on Image and Video Processing* **2007**(3), 4.

Villanueva, A., Cerrolaza, J. J. and Cabeza, R. (2008), *Geometry Issues of Gaze Estimation*, INTECH Open Access Publisher.

Villanueva, A., Ponz, V., Sesma-Sanchez, L., Ariz, M., Porta, S. and Cabeza, R. (2013), 'Hybrid method based on topography for robust detection of iris center and eye corners', *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* **9**(4), 25.

Viola, P. and Jones, M. (2001), Rapid object detection using a boosted cascade of simple features, *in* 'Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on', Vol. 1, IEEE, pp. I–511.

Wang, H., Pan, C. and Chaillou, C. (2009), Tracking eye gaze under coordinated head rotations with an ordinary camera, *in* 'Computer Vision–ACCV 2009', Springer, pp. 120–129.

Wang, Y., Lucey, S. and Cohn, J. F. (2008), Enforcing convexity for improved alignment with constrained local models, *in* 'Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on', IEEE, pp. 1–8.

Watson, G. S. (1964), 'Smooth regression analysis', *Sankhyā: The Indian Journal of Statistics, Series A* pp. 359–372.

Weng, J., Cohen, P., Herniou, M. et al. (1992), 'Camera calibration with distortion models and accuracy evaluation', *IEEE Transactions on pattern analysis and machine intelligence* **14**(10), 965–980.

Wood, E., Baltrušaitis, T., Morency, L.-P., Robinson, P. and Bulling, A. (2016*a*), A 3d morphable eye region model for gaze estimation, *in* 'European Conference on Computer Vision', Springer, pp. 297–313.

Wood, E., Baltrušaitis, T., Morency, L.-P., Robinson, P. and Bulling, A. (2016*b*), Learning an appearance-based gaze estimator from one million synthesised images, *in* 'Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications', ACM, pp. 131–138.

Wood, E. and Bulling, A. (2014), Eyetab: Model-based gaze estimation on unmodified tablet computers, *in* 'Proceedings of the Symposium on Eye Tracking Research and Applications', ACM, pp. 207–210.

Xiao, J., Baker, S., Matthews, I. and Kanade, T. (2004), Real-time combined 2d+ 3d active appearance models, *in* 'IEEE Computer Society Conference on Computer Vision and Pattern Recognition', Vol. 2, IEEE Computer Society; 1999.

Xiao, J., Moriyama, T., Kanade, T. and Cohn, J. (2003), 'Robust full-motion recovery of head by dynamic templates and re-registration techniques', *International Journal of Imaging Systems and Technology* **13**(1), 85–94.

Xiong, X., Liu, Z., Cai, Q. and Zhang, Z. (2014), Eye gaze tracking using an rgbd camera: a comparison with a rgb solution, *in* 'Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication', ACM, pp. 1113–1121.

Yoo, D. and Chung, M. (2005), 'A novel non-intrusive eye gaze estimation using cross-ratio under large head motion', *Computer Vision and Image Understanding* **98**(1), 25–51.

Zhang, X., Kulkarni, H. and Morris, M. R. (2017), Smartphone-based gaze gesture communication for people with motor disabilities, *in* 'Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems', ACM, pp. 2878–2889.

Zhang, X., Sugano, Y., Fritz, M. and Bulling, A. (2015), Appearance-based gaze estimation in the wild, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 4511–4520.

Zhou, R., He, Q., Wu, J., Hu, C. and Meng, Q. (2011), Inner and outer eye corners detection for facial features extraction based on ctgf algorithm, *in* 'Applied Mechanics and Materials', Vol. 58, Trans Tech Publ, pp. 1966–1971.

Zhu, Z. and Ji, Q. (2007), 'Novel eye gaze tracking techniques under natural head movement', *Biomedical Engineering, IEEE Transactions on* **54**(12), 2246–2260.